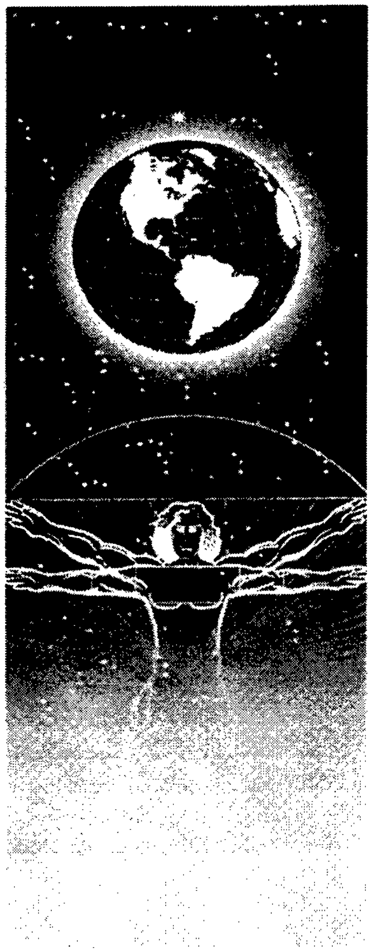
AFRL-HE-WP-TR-2001-0037



UNITED STATES AIR FORCE RESEARCH LABORATORY

Air Force ALP AEF Initiative Wing-Level Cluster Development and Demonstration

Nicholas J. Stute
Christopher S. Allen
Cynthia K. Colby

TASC, Inc.
2555 University Blvd.
Fairborn, OH 45324

Christopher K. Curtis
Air Force Research Laboratory

January 2001

Final Report for the Period February 1999 to August 2000

20010802 043

Approved for public release; distribution is unlimited.

Human Effectiveness Directorate
Deployment and Sustainment Division
Logistics Readiness Branch
2698 G Street
Wright-Patterson AFB OH 45433-7604

NOTICES

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2001-0037

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

ALBERT S. TORIGIAN, Lt Col, USAF
Deputy Chief
Deployment and Sustainment Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 2001	Final - February 1999 - August 2000

4. TITLE AND SUBTITLE
Air Force ALP AEF Initiative Wing-Level Cluster Development and Demonstration

5. FUNDING NUMBERS
C - F41624-99-F-0003
PE - 63106F
PR - 2745
TA - 02
WU - 09

6. AUTHOR(S)
Nicholas J. Stute, Christopher S. Allen, Cynthia K. Colby, Christopher K. Curtis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
TASC, Inc.
2555 University Blvd.
Fairborn, OH 45324

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory, Human Effectiveness Directorate
Deployment and Sustainment Division
Air Force Materiel Command
Logistics Readiness Branch
Wright-Patterson AFB, OH 45433-7604

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-HE-WP-TR-2001-0037

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The purpose of this document is to describe the results for the Air Force ALP AEF Initiative Wing-Level Cluster Development and Demonstration task jointly sponsored by the Logistics Readiness Branch of the Air Force Research Laboratories (AFRL/HESR) and Defense Advanced Research Projects Agency (DARPA). This effort involved the evaluation of the distributed agent based communication and workflow architecture being developed by the DARPA's Advanced Logistics Project (ALP), as it applies to constructing a logistics information system for the deployment of the Air Expeditionary Force (AEF). DARPA's ALP program is researching ways of using intelligent agent technologies to revolutionize the way logistic functions are performed in planning and execution phases of a military operation. To test the ALP architecture, DARPA and other organizations have sponsored various efforts to build applications utilizing the ALP architecture. The AEF Wing-Cluster Development task is one of the many applications being built utilizing the ALP architecture. This task represents the second year of AFRL involvement in the ALP program and includes numerous enhancements to the AEF planning processes that were modeled in the previous effort. The primary focus areas for this effort included dynamic re-planning based on perturbations to the original plan and execution monitoring. The accomplishments in each of these areas are highlighted in this report.

14. SUBJECT TERMS
Advanced Logistics Project (ALP) Air Expeditionary Force (AEF)
Agile Combat Support Clustering Command and Control
Defense Advanced Research Projects Agency (DARPA) Logistics

15. NUMBER OF PAGES
70

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

THIS PAGE LEFT INTENTIONALLY BLANK

PREFACE

This technical report contains the results of the Air Force ALP AEF Initiative Wing-Level Cluster Development and Demonstration task. This work was executed on the TASC's GSA contract, number F41624-99-F-0003. The work described in this report was performed during the period 12 February 1999 through 12 August 2000. The objective of this task was to enhance and expand upon the previous effort of applying the Defense Advanced Research Projects Agency's (DARPA) Advanced Logistics Project (ALP) architecture to model a subset of the logistics operations needed to support the deployment and Sustainment of Air Expeditionary Force units (AEF). The development of the ALP architecture is being executed by the Advanced Logistics Project Integration and Engineering (ALPINE) team. ALPINE is a joint venture between GTE- BBN Technologies and Raytheon Systems.

The principal investigators for this effort included Mr. Chris Curtis, Capt. Adrian Crowley, and Capt. David Sanford from AFRL/HESR, Mr. Nick Stute, Mr. Chris Allen, and Ms. Cynthia Colby from TASC Inc.

Table Of Contents

List Of Figures

ACRONYMS

AEF	Air Expeditionary Forces
AFB	Air Force Base
AFFOR	Air Force Forces
AFRL	Air Force Research Laboratory
AFSC	Air Force Specialty Code
ALP	Advanced Logistics Project
ALPINE	Advanced Logistics Project Integration and Engineering
AMC	Air Mobility Command
API	Application Program Interface
CAMS	Consolidated Aircraft Maintenance System
CINC	Commander In Chief
CORBA	Common Object Request Broker Architecture
DARPA	Defense Advanced Research Project Agency
DDN	Defense Data Network
DLA	Defense Logistics Agency
DOD	Department Of Defense
ERD	Entity Relationship Diagram
FMC	Fully Mission Capable
FOD	Foreign Object Debris
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
IMDS	Integrated Maintenance Data System
JDBC	Java Database Connectivity
J4	Joint Commands Logistics Director
JDK	Java Development Kit
JFC	Java Foundation Classes
JTF	Joint Task Force
LAN	Local Area Network
LEO	Low Earth Orbit
LRC	Logistics Readiness Center
MDS	Mission Design Series
NCA	National Command Authority
NSN	National Stock Number
OPLAN	Operational Plan
ORB	Object Request Broker
PFT	Prepare For Transport
PSP	Plan Service Provider
RMI	Remote Method Invocation
SEAD	Suppression of Enemy Air Defenses
SWA	South West Asia
TRANSCOM	Transportation Command
UI	User Interface
UIC	Unit Identification Code

USAF	United States Air Force
UTC	Unit Type Code
WAN	Wide Area Network
WRM	War Reserve Materiel
XML	Extensible Markup Language

Introduction

The purpose of this document is to describe the results for the Air Force ALP AEF Initiative Wing-Level Cluster Development and Demonstration task jointly sponsored by the Logistics Readiness Branch of the Air Force Research Laboratories (AFRL/HESR) and Defense Advanced Research Projects Agency (DARPA). This effort involved the evaluation of the distributed agent based communication and workflow architecture being developed by the DARPA's Advanced Logistics Project (ALP), as it applies to constructing a logistics information system for the deployment of Air Expeditionary Forces (AEFs). DARPA's ALP program is researching ways of using intelligent agent technologies to revolutionize the way logistic functions are performed in planning and execution phases of a military operation. To test the ALP architecture, DARPA and other organizations have sponsored various efforts to build applications utilizing the ALP architecture. This AEF Wing-Level Cluster Development task is one of the many applications being built utilizing the ALP architecture. This task represents the second year of AFRL involvement in the ALP program and includes numerous enhancements to the AEF planning processes that were modeled in the previous effort. The primary focus areas for this effort included dynamic re-planning based on perturbations to the original plan and execution monitoring. The accomplishments in each of these areas will be highlighted in this report.

Background

For several years, AFRL/HESR has been developing technology to improve logistics operations at USAF operational wings. A central goal of this organization's research efforts is to improve the effectiveness and efficiency of logistic processes at the wing level. This research has ranged from analysis of improved Foreign Object Debris (FOD) control, to technology to provide more environmentally friendly support equipment, to electronic technical orders. AFRL/HESR has been closely following the availability of more powerful computers, and more robust computer connectivity at USAF airbases worldwide. AFRL has commissioned a number of studies as well as actual prototype development efforts to research ways that logistics information could be made available more quickly and accurately to the people needing it. Having this information in a timely manner improves the ability to accurately perform logistics activities that include planning, packaging, shipping persons and materiel to designated operating locations, and keeping those operating locations re-supplied efficiently during the execution phase of an operation.

The ALP program is developing a computer communications infrastructure layer that makes it possible to more rapidly create logistics information systems using a *cluster* architecture. A cluster contains the domain logic required for the logistics operations of an organization. A key to this architecture is the independence of the software application between organizations, so that software can be developed by each organization for its cluster without a tight integration with other clusters representing other organizations. A common application-programming interface (API) between distributed clusters represents the minimum-interfacing requirement that must be adhered to in the development of each cluster. This API provides the communications standards upon which clusters communicate logistic requests to each other, as well as the outcome of those requests. The ALP effort has completed its third year of infrastructure design and development. It has achieved a level of maturity that motivated AFRL/HESR to investigate how this new architecture might be utilized to facilitate the transfer of logistics information, and provide universal access to a variety of tools being developed for wing level support.

The first year of AFRL/HESR involvement in ALP development involved the construction of demonstration software, compliant with the ALP architecture, that modeled a subset of logistic operations at wing level, as a way of demonstrating ALP capabilities. This demonstration software was intended to support two objectives: (1) demonstrate the feasibility of integrating and providing universal access to existing and future logistics tools by utilizing the ALP architecture; and (2) provide a vehicle to demonstrate these ALP based logistics capabilities at the ALP 1998 cluster society demonstration.

As a result of successfully developing an ALP cluster society in the previous effort, AFRL/HESR and DARPA funded this task in order to enhance the previous cluster society and to develop additional functionality in the AEF cluster society. The primary objectives of this effort were to exercise newly available functionality in the infrastructure and to more accurately model wing level logistic processes. As with the previous effort, the ALP society constructed under this effort was part of the annual integrated ALP demonstration in January 2000.

To demonstrate the potential capability that an ALP USAF wing cluster would provide, it was decided by the team that a scenario would be needed that involved multiple wing units, as well as other units typically involved in logistics processes. It was also decided that it would be desirable to have several wing clusters interact with external organizations, also represented by clusters, in order to highlight the capabilities of the architecture in the ALP 1999 demonstration. The established scenario involved the short-notice deployment of two Air Expeditionary Force (AEF) units bedding down in two theaters of operations within 72 hours of strategic warning.

1999 Scenario

The scenario developed for the 1999 ALP demonstration was a product of iterative brainstorming by the government-contractor team. The team started with the scenario developed last year and modified it in order to more realistically represent how AEFs are utilized and planned. The Air Force has been organizing AEFs for rapid reaction deployments for several years. This concept, which is still evolving into the Expeditionary Air Force, provides for specific flying units to be maintained at a high state of readiness for immediate employment. To obtain the required responsiveness, information must flow quickly between the AEF, its constituent units, Joint Task Force (JTF) components, in-theater bases, War Reserve Materiel (WRM) locations, Transportation Command (TRANSCOM), and Air Mobility Command (AMC). The following steps outline the activities that were modeled using the ALP architecture during a tasking of the 2 AEF units. The demonstration constructed during this effort commenced with step 6.

1. National Command Authority (NCA) authorizes the Commander In Chief (CINC) to execute an operational plan (OPLAN).
2. CINC forms JTF.
3. OPLAN formed by JTF.
4. Logistics plan formed by JTF, coordinating with operations plan.
5. Logistics organizations are notified of logistics requirements to support OPLAN.
6. One of the top-level logistics organizations notified is the Joint Commands Logistics Director (J4) in the Air Force Forces (AFFOR) component of the JTF.

7. The AFFOR J4 notifies the J4 component of the active AEF units in the rotation of the start of execution of the specified OPLAN.
8. The AEF notifies its components of their specific missions including the number of aircraft to deploy and asks them to provide their logistics support requirements (equipment and personnel).
9. AEF units compile their requirements and forward the details back to the lead wing.
10. The lead wing combines, consolidates, and tailors all of the logistics requirements received from its deploying squadrons.
11. The lead wing uses multiple servicing techniques to select the "lease expensive" items for transport, comparing availability of assets in theater, at WRM locations, and from CONUS sources.
12. Transportation requests for airlift to transport sourced items are sent to TRANSCOM from each organization supplying assets for the deployment.
13. TRANSCOM provides airlift schedules back to the wings.
14. Sustainment requests for fuel and critical spare parts are generated and sent to DLA.

Upon completing the scenario, the contractor-government team identified the AEFs that would be needed to satisfy the scenario. It was determined that AEF1, AEF2, and AEF9 would be modeled as a series of ALP clusters. The scenario was to take place in southwest Asia (SWA) so the team needed to identify hypothetical OCONUS wings where the deploying AEFs would beddown. For the scenario, the following hypothetical wings were utilized: Aviano Airbase and Prince Sultan Airbase. After defining the scenario and the organizations that would be represented, the team embarked on developing the clusters representing each of the organizations and their associated logic required to ensure sufficient logistics support for completion of their missions.

Background for this Year's Scenarios

The ALP community identified several objectives and goals for the overall ALP project to be accomplished during the 1999 development effort. Also, specific AEF development initiatives were identified to further enhance the completeness and robustness of the AEF capabilities modeled in ALP for this year's efforts.

In the previous ALP development effort, the primary goal was to develop the logistics plans to support the various missions identified in a representative OPLAN. This year additional goals include:

1. Extend the society with both depth and breadth. That is, increase the number of organizations that are represented in ALP for the existing services (Army, Air Force), and begin to create ALP models for organizations in the Navy and Marines.
2. Go beyond the initial planning phases to incorporate replanning capabilities. This includes execution monitoring, reacting to real-world events, replanning due to changes in operational needs or objectives including OPTEMPO, and Sortie Rate changes.

Specific goals for the AEF community include:

1. Enhance functionality to capture current vision of AEF deployment processes.
2. Support the ability to deploy two simultaneous AEFs.
3. Extend the functionality of the automated selection of aircraft to be usable by Marine squadrons.
4. Identify and develop AEF scenarios to demonstrate ALP's ability to automatically replan in response to execution events and/or perturbations in OPLAN elements.
5. Add functionality to support selection of personnel.

A later section will discuss this year's efforts to accomplish these goals. First, the ALP infrastructure will be discussed followed by details on the new functionality added to the AEF society under this effort.

ALP Architecture

The general ALP architecture has as its cornerstone the assumption that every organization will have a standard infrastructure for communicating logistics information. Logistics information in this context includes large amounts of messages and data, including requests for equipment, supplies, and ammunition, shipping/airlift schedules, inventory status, logistics readiness status, and airbase beddown conditions. In the past, such information requests, and replies to those requests, either had to pass through many different information systems or be made by voice communications. The former method, requiring multiple information systems, is a result of the historical independent development of systems to solve parts of the logistics information transfer challenge, without acknowledging a vision of the comprehensive logistics processes.

The technology available today has improved to the point that the computers available on logisticians' desktops are capable of performing at a much higher level. These computers are typically now networked to computers around the world via the Defense Data Network (DDN) and the Internet. In fact, this is the key enabling infrastructure capability that will make ALP processes feasible. With a reliable data communications network in place, much of the information exchange that previously required person-to-person communication can now be done automatically between networked logistics information servers and their clients. Today, these network links are present through high-speed dedicated landlines and telephone networks. In the very near future, high-speed capabilities should be widely available even in remote areas through the new constellations of Low Earth Orbit (LEO) satellites being set up by multiple telecommunication providers. Although security issues are still being addressed, it is clear the available bandwidth this technology provides would radically improve Department Of Defense (DOD) information distribution capabilities.

Key Features of the ALP Architecture

The ALP system is a highly automated system. Logistics decision-making logic, the capability to assign equipment and personnel, and the capability to schedule resources and transportation are programmed into the ALP system components. Although automated, users have an important role in the executing ALP system, users provide the policies and rules that govern these processes, can approve decisions made by the system, intervene when necessary to resolve conflicts, and provide solutions for shortfalls and issues with time constraints. Users can also inject tasks into the system for actions that may not have been considered by the system designers, or to incorporate real world events into the processes.

The ALP system is a highly distributed system comprised of many *clusters*. Recall that a cluster is a portion of the "society" representing the domain logic of a particular organization, such as TRANSCOM, Defense Logistic Agency (DLA), or perhaps a maintenance group or a base hospital. The scope of a cluster's functionality is quite flexible. A cluster's functionality can be specific to a small sphere of activities or processes, such as the assignment of housing at a base. Or, the functionality may be more encompassing, such as the entire process of deploying a combat mission. The collection of all ALP clusters is called the ALP *society*. A subset of clusters makes up a *community*. For example, the set of clusters that are created to represent the functionality at a Fighter Wing could be considered a community. Note that a given community could have a great deal of processing that is done independently from the ALP society as a whole, e.g., allocating and scheduling resources for training missions, periodic maintenance, food supplies, etc. The same community could also be involved in a society-wide scenario, such as the deployment of forces and equipment for a particular mission.

ALP provides the definition of standard communications protocols that sit on top of the network hardware infrastructure. Clusters provide different types and levels of information, but the ALP architecture ensures that the "clusters speak the same language" so they can exchange information or services.

Based on having a standard set of communication messages between clusters, groups of ALP clusters can be set up to operate with each other, sharing information seamlessly, and requiring very little human intervention to facilitate the communication process. Information, which currently may take several phone call inquiries to different USAF installations, would, in most situations, flow automatically to the clusters requiring it. Automatic updates of data would be provided as real world situations change. The potential seamless and near real time dissemination of messages between clusters could provide the logistician with a more complete up-to-date picture of the state of an operation or the planning of an operation that is currently available.

<u>The ALP system provides continuous replanning and updating</u>. The status of real world events can change the availability of resources, or can change the priority of future events. With the ALP system, the availability or status of resources can be monitored and allocated/reallocated to improve timeliness, cost, effectiveness, minimize loss of life, or similar uses. These changes can be handled by cluster functionality, resulting in changes to allocated resources, or can cause elements of the cluster's LogPlan to be changed or new elements added. The ALP infrastructure automatically propagates these changes to other clusters, where subsequent plan alterations may

be initiated. Moreover, ALP supports the combination of planning and execution requirements. As time passes, planned events become reality, and then become things of the past. The results of those events can have an effect on future events. The policies and functionality to support this replanning can be incorporated into the logic of the clusters.

ALP Cluster Concepts

Figure 1 shows the basic components of a cluster. The ALP API defines the methodologies for clusters to communicate with each other and the methodologies for each cluster to communicate information within itself. A cluster makes requests of other clusters via outgoing directives and receives requests from other clusters via incoming directives. Different clusters can be resident on the same machine or reside on different machines. Note that the words "directive" and "task" are used interchangeably in this document and in the ALP architecture documentation.

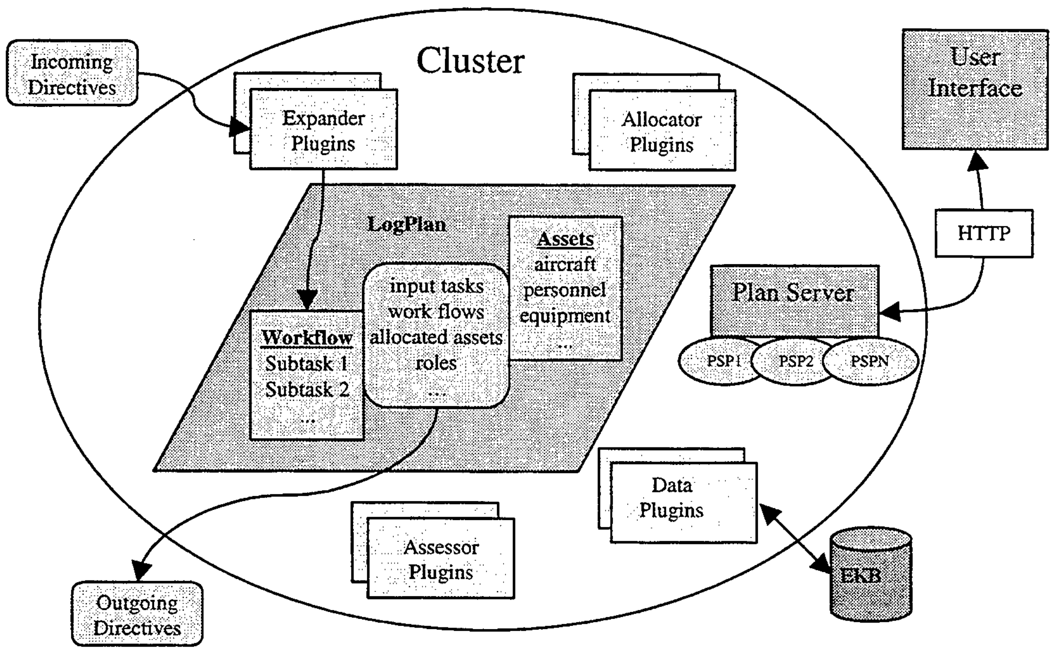


Figure 1: Cluster Components

A cluster consists of ALP LogPlan elements, a combination of various types of *plugins*, and, in order to fit into the society, portions of the ALP infrastructure. Plugins are modular, self-contained discrete units that provide the domain specific functionality of the cluster. It is intended that, as functionality is created, it can be "plugged into" the cluster, giving the cluster the ability to handle more tasks. As functionality is maintained (fixes, enhancements, etc.), plugins can be "pulled out" and replaced with updated ones, without interrupting the rest of the system's processing.

There are several types of plugins. These include expander, allocator, assessor, data, and user interface (UI) plugins. The following sections provide a description for each of the plugin types.

Expander Plugin

An expander plugin, also called a task expander, performs the initial processing of each input task received by the cluster. This plugin expands input tasks into one or more subtasks that the cluster knows how to complete. Each input task's set of subtasks is referred to as a *workflow*.

For example, the input task "Generate AEF using OPLAN10A" might be expanded into a workflow containing the following subtasks:

- Subtask 1 = "Determine requirements for Suppression of Enemy Air Defenses (SEAD) using OPLAN10A"
- Subtasks 2 = "Determine requirements for Air Interdiction using OPLAN10A"
- Subtasks 3 = "Determine requirements for Air Superiority using OPLAN10A"

Allocator Plugin

A cluster's assets can include both physical assets and other cluster assets. It is the responsibility of an allocator plugin to allocate the cluster's assets to complete the subtasks of each workflow. The allocator plugin may also choose to delegate the responsibility for completing a subtask to another cluster, which is the principal means of creating an output directive. The plugin would choose the target cluster for this kind of directive by means of inter-cluster *relationships* and cluster *capabilities or roles*. These concepts are discussed later in the section titled "Cluster Relationships."

It is the allocator's function to maintain the "best" allocation of the cluster's assets. That is, as the plugin considers assets for new workflows, it may be necessary to reallocate existing asset allocations in order to improve the overall usage of its resources. For example, suppose an allocator plugin has designated a particular truck to perform a transportation request. Further, suppose the cluster receives another transportation request. It may be more economical to deallocate the first truck and allocate a single larger one to handle both transportation requests, than to allocate a second truck dedicated solely to the new transportation request. This type of logic would have to be built into the allocator plugin of the cluster.

In addition to allocating the cluster's assets, it is the allocator's responsibility to assign schedules of usage and *penalty values* for its allocations. Penalty values represent the costs associated with each allocation, where a cost could be in terms of dollar value, some risk value, a hardship factor for giving up the asset, etc. The penalty value may also be a combination of these things. This area of the ALP architecture has the potential for further expansion and high fidelity based on real world statistical information related to the cluster's business activities.

Assessor Plugin

An assessor plugin is responsible for monitoring the execution of the plan. By considering real world events, including satisfactory completion of projected plan components, as well as verifying overall objective conformance, an assessor can watch for plan deviations. The plugin may incorporate various thresholds to ensure that successful plan execution is not jeopardized. Assessor plugins can generate exceptions to alert appropriate mechanisms that remedial actions may be required or may simply insert new tasks into the system to directly effect replanning processes.

Data Plugin

Data plugins are responsible for mapping contemporary data into the ALP society. This is the means for providing an ALP wrapper for existing data sources. It is the data plugin's responsibility to maintain interfaces with its data sources and to act as a liaison between ALP processes and the processes that natively work with each of its data sources. This could include updates to contemporary databases due to ALP processing or may include updates to ALP processing due to triggers set up in the databases.

User Interface (UI) Plugin

UI plugins are created to provide the users with views of ALP's decision-making processes and the plan elements that ALP generates. UI plugins also provide the user with the ability to approve or modify the decisions that have been automated by ALP processes. Where applicable, the UI can allow the user to enter or modify rules that govern actions and decisions for plugin processing.

Plan Service Provider (PSP)

The PSP mechanism is a new feature that was added to the architecture during this effort. The primary design goal of the PSP mechanism was to expose the contents of the log plan through a network protocol for purposes of creating user interfaces. A hypertext transfer protocol (HTTP) server (typically referred to as a web server) resides on each cluster. The user creates PSPs, which are accessed through the web server running on the cluster. Any client needing information about the cluster can communicate with the PSP using standard HTTP protocol to query for specific LogPlan information. The PSP has most of the same privileges as a normal plugin; in particular it can query the contents of the log plan. The PSPs are written to produce either HTML or XML or both.

Utilizing the PSP mechanism, user interfaces can be created which run separately from the cluster society. Essentially, the user is writing a web application when using the PSP mechanism. For the AEF society, a version of the squadron user interface and a transportation user interface utilize the PSP mechanism. The PSPs developed for these user interfaces only generated XML.

LogPlan

Consider the section in Figure 1 labeled "LogPlan." Each ALP cluster contains a LogPlan that reflects the processes and planning accomplished by that cluster. In actuality, this is the collection of all of the cluster's data including, among other things, the cluster's assets, input tasks, workflows, its relationships with other clusters, and the actual logistics plan elements. Note that each cluster in the ALP society maintains its own LogPlan. The LogPlan maintains the state of a cluster. The capability to persistently store LogPlans was developed by ALPINE during the 1999 ALP architecture development. Java's serialization mechanism was leveraged for implementing the persistence of the LogPlan.

All of the plugin types communicate directly with the cluster's LogPlan. Depending on the situation, this link may be for read-only functionality. For example, certain UI plugins may be created to display various views of the logistics information, such as asset usage, or schedules. On the other hand, when the expander plugin creates a workflow during a task expansion, it writes the workflow directly to the LogPlan. The allocator plugin finds new workflows and available assets in the LogPlan and submits asset allocations, schedules, and penalty values back to the LogPlan.

Cluster Relationships

Clusters form *relationships* with each other, establishing *capabilities* and *roles* in the process. At a minimum, each cluster is required to have a "superior" cluster. The exception to this is the one cluster that resides at the top of the cluster hierarchy. During the cluster's startup phase, ALP establishes a Superior/Subordinate relationship between the cluster being initialized and its superior. Then, while the clusters are processing, each one will have a reference to the other in its list of Organization assets. Moreover, there will be a role associated with the reference, in this case, either "superior" or "subordinate." In addition to this automatically generated relationship, clusters can selectively establish relationships and roles with each other. These relationships can also include capabilities. For example, cluster A may have cluster B as a "supporting" cluster with "division supply provider" capability. In this example, B has a role of "supporting" cluster A, and A sees B as having the capability of "division supply provider." When cluster A's allocator is assigning resources to subtasks, it might decide to redirect subtasks to cluster B when a "division supply provider" allocation would be appropriate.

Putting the Pieces Together

Figure 2 gives a linear presentation of plugin activity as a cluster and its plugins process a single task. After a cluster receives a task, it is passed to the expander plugin. The expander creates a workflow of subtasks and submits it to the LogPlan. The allocator plugin gets the workflow of subtasks, finds assets to allocate to them (or perhaps assigns tasks to another cluster), calculates penalty values, and submits the results to the LogPlan. The ALP infrastructure creates notification information to pass back to the cluster that made the original request. An assessor plugin may also review the results of the allocations and generate additional directives.

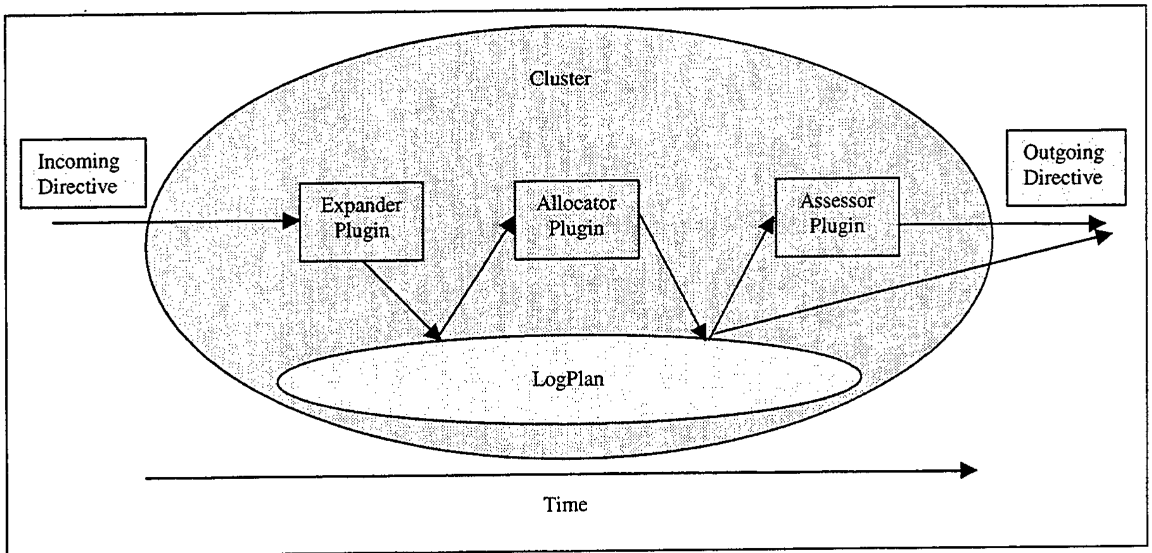


Figure 2: Plugin Operation Flow

ALP's Decision-Making Philosophy

ALP is based on a decision-making philosophy focused on providing solutions that continually improve tolerances rather than attempting to initially provide a "best" solution. The motivation for this is the highly distributed nature of the ALP architecture. To achieve a "best" solution would require a centralized location to request everything of every provider, then decide for everyone, who gets what and when. But, in a highly distributed system, a centralized location containing all the rules and logic does not exist. Even if it did, the amount of data required would be prohibitive. ALP provides a different solution. Each task is created with an associated *penalty function*; a penalty function can be thought of as a set of thresholds. Recall from the previous section that an allocator supplies a penalty value when it allocates a resource. The requester of the task will get that allocation's penalty value and will pass it to the task's penalty function. If the penalty function indicates the penalty value is "acceptable," the cluster could just accept the situation and continue. If it is "unacceptable," the cluster could rescind the task and request a different cluster. This assumes there are other clusters available to do the task; otherwise, "unacceptable" may necessarily be accepted. If the penalty function indicates a "borderline" condition, the cluster could keep the allocation but start creating additional tasks to do some comparative shopping. If the cluster finds a more acceptable allocation from a different source, it could keep the alternate allocation and rescind the original request.

Note that this methodology focuses on keeping all of the elements of a plan within tolerance levels. This approach vastly reduces the number of requests that have to be passed from cluster to cluster, resulting in fewer burdens placed on the communications processes. Also, note that this is where assessor plugins can play an important role. These plugins could generate low-priority requests that are intended to find alternative solutions to improve tolerance levels, but could be processed during "lower" activity times.

ALP Development Environment

The ALP architecture development team elected to use Java and Java-based tools for the development of the ALP architecture. Java provides the platform independence, and includes powerful networking capabilities as a part of the language. The current release of the ALP architecture utilizes Sun's Java Development Kit (JDK) version 1.2.1.

The development team for this effort also utilized Inprise's Java integrated development environment (IDE) named JBuilder 3 for constructing the demonstration software. Pentium II/III -based personal computers running Microsoft's Windows NT operating system were the development machines used. Although a personal computer/Windows configuration was utilized, cross platform capability was accomplished as a result of doing all development utilizing the Java programming language. The demonstration software was successfully executed on a Linux platform as well.

ALP Infrastructure and Architecture Enhancements for 1999

Most of the previous ALP Infrastructure and Architecture discussion relates to concepts and mechanisms that have been in place for quite some time. During the 1999 development cycle there were considerable enhancements to ALP features as well as changes to methodologies for developing effective plugins. These features include PSP mechanisms, Alerts/Triggers, ALP simulation time, and remote method invocation (RMI).

The discussion of many of these enhancements is beyond the scope of this document. For a complete description of the state of the ALP infrastructure refer to the documentation available on the Cougaar web site (http://www.cougaar.org). One of the objectives of the ALP community was to make available the ALP architecture to the general public. To accomplish this, the Open Source model was utilized. The ALP architecture was packaged such that the core architecture was isolated from any specific military logistic functionality. The newly packaged architecture was renamed COUGAAR and is available on the Cougaar web site.

As discussed in the Key Features of the Alp Architecture section, the PSP mechanism, Alerts/Triggers, and ALP simulation time were all utilized in the enhancements added to the AEF society under this effort. The PSP mechanism provides a HTTP protocol interface into the log plan for developing user interfaces.

To support demonstrating dynamic replanning and what if planning ALPINE introduced the concept of ALP simulation time. The simulation time mechanism provides the functionality to associate the actual time with simulation time. For example, the user can set up a society so that for every minute of elapsed time results in the alp simulation time to advance by one day. A user interface was developed by the team to allow the user to set the amount of simulation time advanced for some increment of real time. The simulation time mechanism was utilized by the fuel assessor plugin and one of the perturbations developed.

1999 AEF Society

This section describes the 1999 AEF Society and describes the new features that were developed to address the objectives and goals that were mentioned in the section "Background for this Year's Scenarios".

The following list provides a high level summary of the major additions and enhancements added to the AEF society during this effort. The sections following will provide further detail on each items listed.

1. More detailed representation of the organizations involved in the AEF deployment process. This includes modeling the following as clusters: wings, squadrons, beddown

bases, and command organizations. Details of the organizations modeled in the AEF society are discussed in the section titled The AEF Standalone Society.

2. More detailed modeling of the AEF vision of AEF deployment processes.
3. As a result of a much more detailed representation of the AEF deployment planning process, much more data was needed. The team switched from using flat files to a relational database in order to efficiently store and retrieve the needed data to support the AEF cluster society. Details on this part of the effort are defined in the AEF Databases section.
4. Support dynamic replanning as a result of deviations from the original plan. Three different dynamic replanning scenarios were supported in the AEF society during this year's effort. Details on this effort are described in the AEF Perturbations section.
5. Interaction with the DLA cluster community for handling requests for fuel and spare parts sustainment tasks. Details on this effort are described in the DLA Interaction and Integration section.
6. Enhanced interaction with the TRANSCOM cluster society including a response mechanism from the TRANSCOM cluster society back to the AEF society. Details on this part of the effort are defined in the TRANSCOM Interaction and Integration section.
7. Starting with the interfaces developed during the previous effort, the team added additional user interfaces and enhanced of the existing ones. Details on the user interfaces available in the AEF society are discussed in the AEF User Interface section.

The AEF Standalone Society

The team created a standalone society of clusters to demonstrate the generation of an Air Expeditionary Force (AEF). Figure 3 shows the clusters that are included in the demonstration. A stub cluster was created for the standalone society to emulate the TRANSCOM and DLA cluster communities, and non-cluster functionality was developed, called JFACC, for the sole purpose of launching the AEF generation scenario. The Air Force clusters and plugins are the same in the standalone version as those used in the full DARPA society. The only difference between the standalone society and the complete integrated ALP Demonstration society was the stub clusters representing DLA and TRANSCOM were not used.

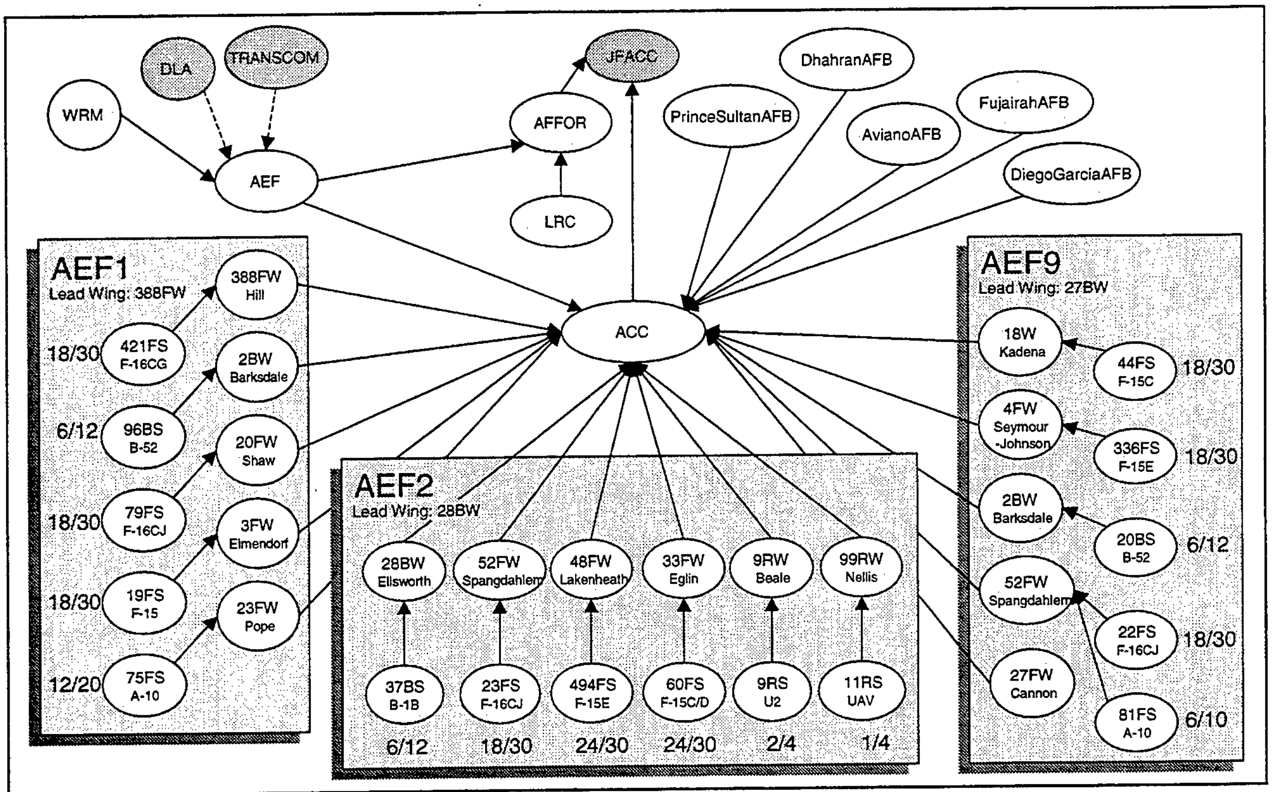


Figure 3: Air Force Demonstration Society

Three of the 10 AEFs were modeled in the AEF society, these included AEF1, AEF2, and AEF9. For each AEF modeled, a cluster was created for each squadron and wing making up the particular AEF. The same set of plugins was used to create each of the squadron clusters. Similarly, the same set of plugins was used to create each of the wing clusters. The only difference in each of the squadron and wing clusters was the data in the database that identified the assets available at each cluster.

The squadron clusters contain the domain logic for intelligently selecting aircraft for the planned deployment. Historical data for each of the aircraft was modeled in a relational database. When one of the squadron clusters was tasked to plan for the deployment of some number of aircraft, it would consult the historical data on its aircraft to select the "best" for the deployment. This selection process takes into consideration past performance of each aircraft as well as future scheduled maintenance. The user also has the ability to change characteristics of the aircraft selection decision logic through the Squadron user interface. Details on how to accomplish this are discussed in the AEF User Interface section. The squadron clusters also modeled support

equipment and personnel assets. All of the information identifying what assets and personnel are available is stored in the database and accessed when the society is run. The following squadron clusters were created in the AEF cluster society: 421st Fighter Squadron, 96th Bomber Squadron, 79th Fighter Squadron, 19th Fighter Squadron, 75th Fighter Squadron, 37th Bomber Squadron, 23rd Fighter Squadron, 494th Fighter Squadron, 60th Fighter Squadron, 9th Reconnaissance Squadron, 11th Reconnaissance Squadron, 44th Fighter Squadron, 336th Fighter Squadron, 20th Bomber Squadron, 22nd Fighter Squadron, and 81st Fighter Squadron.
Each of the squadrons belongs to a wing organization that is also modeled in the AEF society. For each AEF, there is a wing identified as the "Lead Wing". Similar to the squadron clusters, the wing clusters also have physical assets available to support the deployment process. The functionality available in the wing clusters include supplying assets, processing prepare for transport tasks from the TRANSCOM cluster community, and consolidating UTCs. The consolidation logic is only used in the lead wing clusters. The following wings were modeled in the AEF society: 388 Fighter Wing, 2 BW, 20 Fighter Wing, 3 Fighter Wing, 23 Fighter Wing, 28 Bomber Wing, 52 Fighter Wing, 48 Fighter Wing, 33 Fighter Wing, 9 Reconnaissance Wing, 99 Reconnaissance Wing, 18 Wing, 4 Fighter Wing, 2 Bomber Wing, 52 Fighter Wing, and 27 Fighter Wing.

Five beddown clusters were created to support the scenario. These clusters included Prince Sultan Air Base, Dhahran Air Base, Aviano Air Base, Fujairah Air Base, and Diego Garcia Air Base. These clusters shared many of the plugins that made up the wings defined previously. One of the distinguishing features of the beddown clusters is the responsibility to create tasks to send to the DLA cluster community for sustainment requests. Details on this aspect of the beddown clusters are described in the DLA Interaction and Integration section.

A collection of command clusters were also created whose primary functionality was to provide the conduit of the high level tasks to flow through the society. The JFACC cluster was created to act as the starting point for the AEF cluster planning process. Its sole purpose was to create the first task in the system to get the planning process started. Similarly, the AFFOR, ACC, and LRC clusters were created to represent the true command and control mechanism, and provide a very basic level of functionality.

Air Force Standalone Society – Task Flow

The ALP infrastructure defines a pseudo-grammar to be used to construct tasks. This pseudo-grammar consists of objects, verbs, direct objects and prepositional phrases. Although the various components of each statement are stored separately in Java class elements, the flow of tasks can be described using simple prose. Following is a description of the flow of tasks created during the demonstration. Each task will be highlighted using an italic font.

The JFACC cluster launches the demonstration scenario by sending the *Get Log Support* task to the ACC cluster. The expander plugin in the ACC cluster is activated since a new input task was received. From the new input task, ACC's expander plugin creates a workflow containing a single subtask: *Execute OPLAN*, which it sends to the AFFOR cluster. The AFFOR cluster's expander plugin is activated since a new input task was received. From the new input task, the AFFOR expander creates a workflow containing two subtasks: *GenerateAEF For AEF1* and *Generate AEF For AEF2*. These subtasks use nomenclature that is certainly not normal English, but is typical in communicating requests between ALP clusters. In this subtask, "GenerateAEF" acts as the verb; "For AEF1" is a specific classification of which AEF is to be deployed. The two GenerateAEF subtasks are sent to the AEF cluster. Note that the AFFOR cluster could not just make up such a request and expect the AEF cluster to magically know how to act upon it. Rather, each cluster, the AEF in this example, publishes the formats of statements that it knows how to process. Then, clusters can create statements that conform to that format and can send them to the cluster that published them.

The AEF cluster receives the two tasks and its expander creates *GenerateAEF1* and *Generate AEF2* tasks and routes them to the respective lead wings for the identified AEFs. The AEF clusters determine the respective lead by information specified in the OPLAN. In this case the *GenerateAEF1* task is routed to the 388FW cluster (lead wing for the AEF1) and the *GenerateAEF2* is routed to the 28 Bomber Wing (lead wing for the AEF2). From this point on, there are two threads of tasks generated that are mirrors of each other. There is a thread that supports planning for AEF1 and a thread that supports planning for AEF2. Only the AEF1 planning thread is discussed, but the same functionality is employed for the AEF2 thread. Figure 4 provides a pictorial view of how the high level tasks flow through the AEF society.

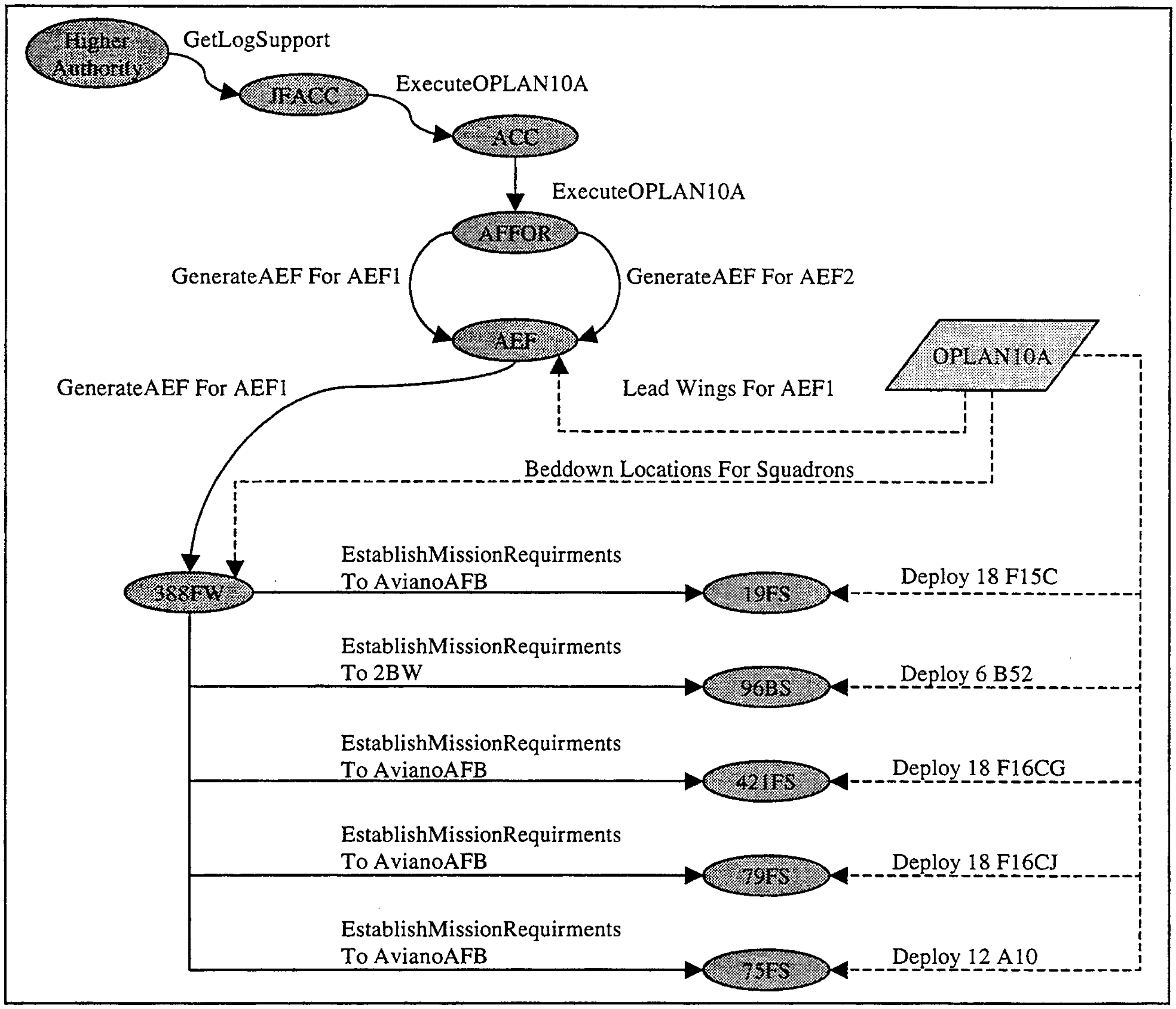


Figure 4: Overall High Level Task Flow

The 388FW cluster receives the *GenerateAEF For AEF1* task from the AEF cluster and then creates a series of subtasks. Based on information obtained from the OPLAN, the 388FW determines the squadrons and their beddown destination in support of the AEF. From this information, the 388FW generates and sends *EstablishMissionRequirements* tasks to each of the squadrons. On each of these EstablishMissionRequirements tasks is a To clause indicating the beddown location (See Figure 4). After receiving the EstablishMissionRequirements task the squadron clusters consult the OPLAN content to determine the quantity of aircraft needed to be deployed. For example, the 79FS determines the OPLAN calls for 18 F-16CJ aircraft to be supplied.

Once the squadron determines the quantity of aircraft to be deployed, it generates several subtasks associated with the *EstablishMissionRequirements* input task (see Figure 5). A *ProcessUTC* task is created and routed back to the lead wing cluster (388FW). This *ProcessUTC* task has attached to it a data structure representing the standalone equipment UTC needed to support the number of deploying aircraft. Also dependent on the number of aircraft deploying is the amount and type of personnel needed to support the deployment; this information is captured in the *ProcessPersonnelUTC* task. The *ProcessPersonnelUTC* has attached to it a data structure that identifies the type and quantity of personnel needed. This task is routed back to the squadron that created it. If the squadron is going to beddown at a base other than its home base, a *GenerateBeddownSupport* task is created and routed to the beddown cluster. The *GenerateBeddownSupport* task notifies the beddown location on the quantity of aircraft and personnel that will be arriving so it can plan to support it. Note, that in the scenario, the bomber squadrons are not relocated to in-theater bases so no *GenerateBeddownSupport* task is generated. The final task created is the *SelectDeployment*. The *SelectDeployment* task identifies the quantity of aircraft and the beddown location for the aircraft. This *SelectDeployment* task will result in the aircraft being selected and allocated for the deployment. Similarly, the *ProcessPersonnelUTC* task results in the personnel being selected and allocated for the deployment. A more sophisticated logic is employed to allocate the assets needed to satisfy the equipment UTCs.

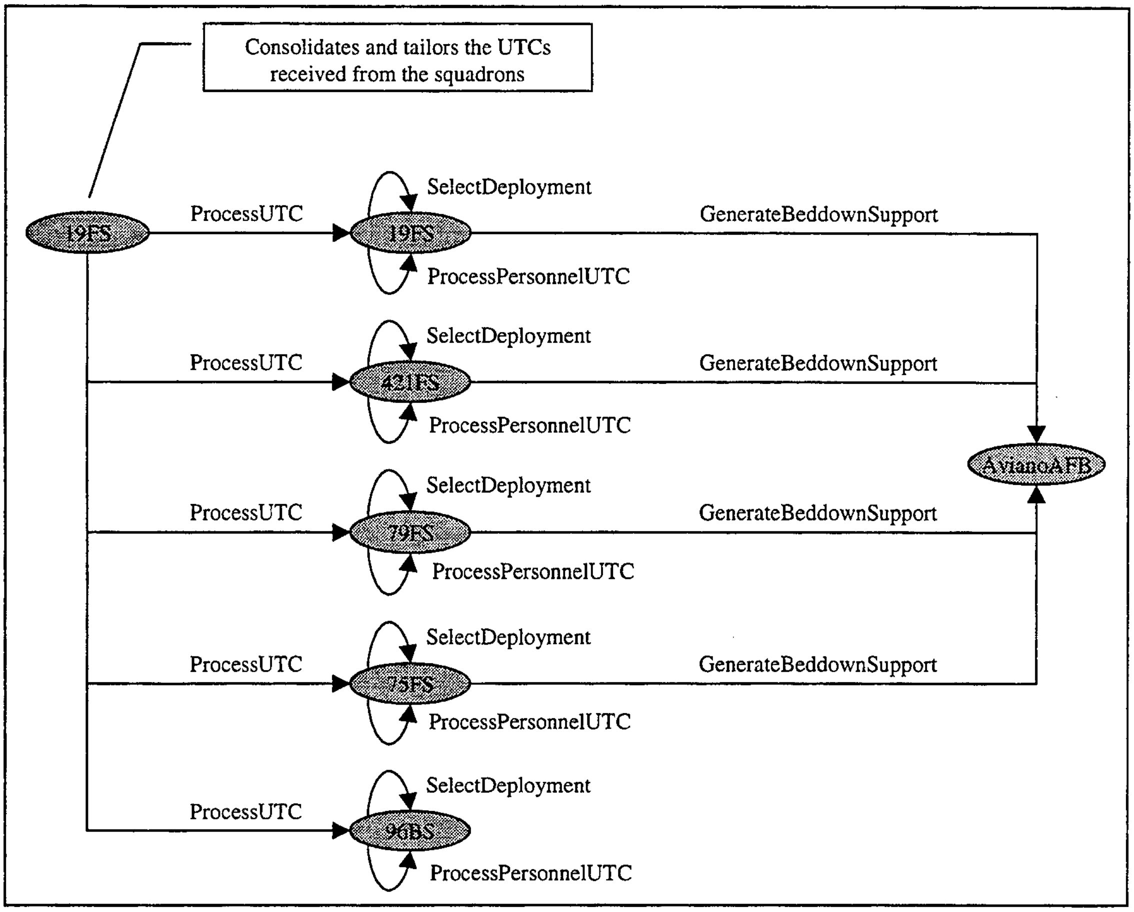


Figure 5: High Level Task Flow

Upon the 388FW receiving all of the *ProcessUTC* tasks back from the squadrons, the cluster will consolidate and tailor the UTCs. The consolidation and tailoring rules are specified using the Rule Based Tailoring View, described in the User Interface section. After the consolidation and tailoring logic has been applied, the Source/Supply logic unique to the AEF society is utilized to allocate the assets needed to satisfy the tailored UTCs.

TASC has developed a set of ALP capabilities that support multiple suppliers in a Source and Supply chain. This is a feature that is not available from the ALP infrastructure. Using this functionality, developers can identify multiple suppliers for supply tasks. The supplier list can also be ordered, indicating a preferred order of use. Using the allocation results obtained from each supplier, the plugins iteratively make requests of the alternative suppliers.

For example, suppose an organization needs 10 Power Carts. The cluster representing that organization creates a SOURCE task requesting these 10 Power Carts. This request is sent to the first of its suppliers. If that supplier can supply 6 of the desired 10, the plugins then create an additional subtask requesting the remaining 4 Power Carts. This new subtask is sent to the second supplier. This process continues until all items are supplied or until there are no more suppliers.

The functionality developed to support these features is also responsive to planning-time and execution-time changes. In particular, if a supplier has promised to supply a quantity of items but later reduces part or all of that quantity, the source/supply functionality will revisit the capabilities of all of its suppliers to try to fill the new shortfall.

The ALP development community has expressed a strong interest in the generalization of these features into a set of plugins that perform the above (and some additional) functionality. TASC plans to develop these plugins in a follow-on development effort.

Once all the allocations have been made to satisfy the equipment UTCs, transportation tasks need to be sent to the TRANSCOM cluster in order to schedule transportation to move the asset and personnel to the beddown location. Recall at the beginning of this discussion, that the AEF cluster created a *GenerateAEF* task and routed it to the 388FW. The AEF cluster will receive an allocation result once this task completes, at which point all the planning for deploying the AEF has completed. Upon receiving this notification, the AEF cluster creates a series of *GenerateTransportation* tasks to all of the organizations that supplied assets or personnel in support of the deploying AEF. Once an organization receives the *GenerateTransportation*, it will create a *Transport* task and route it to the TRANSCOM cluster. Associated with this *Transport* task, is a data structure that identifies all of the assets needed to be transported and where they need to be transported.

Penalty Values in the Air Force Society Demonstration

The wing, squadron, beddown, and WRM clusters in the demonstration are used as resources to supply the assets required to support the AEF deployments. As these clusters allocate assets to satisfy requests, they must calculate and assign a penalty value. The penalty value assignment is intended to capture a transportation factor and an operational factor. Assets at the beddown clusters have a transportation factor of zero and WRM assets have an operation factor of zero. As a result, the demonstration scenario ultimately sources each asset from the beddown location until the operational cost surpasses the transportation costs at the WRM. Then WRM sources assets as long as it has assets available. Finally, the wing and squadron clusters supply the assets. Note that this makes the (perhaps-oversimplified) assumption that WRM sources have a lower transportation penalty value (cost) than those from the fighter wing sources.

AEF Perturbations and Dynamic Replanning

One of the focus areas for the ALP project this year was to support dynamic replanning. To demonstrate this functionality within the AEF society, the team defined a set of perturbations that the AEF society would respond to and replan against. The team added support in the AEF society for three different perturbations. The main objective of the perturbations was to have the society automatically recognize the perturbation and dynamically replan if necessary as a result of the perturbation.

The 1999 scenario called for the deployment of two AEF units. Some time during the execution of the scenario, it was determined that an additional AEF would need to be deployed. In order to support this, the AEF society needed to react to the perturbation of the request for an additional AEF to be deployed. To accomplish this, a plugin was created that was attached to the JFACC cluster. This plugin reacted to a task indicating the need for an additional AEF to be deployed. When this task was received, the JFACC plugin altered the original OPLAN to indicate the fact that AEF 9 needed to be deployed. Once this change was added to the OPLAN, the same tasking process used to support deployment AEF 1 and AEF 2 was utilized. The UI section describes how to activate this perturbation in the AEF society.

Another perturbation supported in the AEF society is the ability to replan based on changes in the capacity aspect of assets. Each asset in the AEF society has an attached property that models its capacity. The team used this property to represent the condition of the asset. A simplistic approach was taken in that the asset either has a 100% capacity or a 0% capacity. An asset with a 0% capacity was defined as unusable. To support this perturbation, an assessor plugin was created that monitored the capacity aspect on assets. If the assessor plugin determined that an asset's capacity changed to 0%, it would trigger the cluster to replan if the identified asset was involved in support of the deployment of any of the AEFs. This replanning involved attempting to get another asset supplied to replace the one whose capacity went to 0%. First the cluster that made the allocation will determine if it has another of the same type of asset to allocate. If not, the cluster will ask its suppliers to allocate the asset. If this is unsuccessful, the cluster will then report back to its superior by changing the quantity on the reported value. The superior will then attempt to supply the asset itself and if not successful it will request its suppliers. This will continue until there are no more superiors to task. If the ultimate superior cannot get the asset supplied, it will result in a shortfall.

The following steps describe how to reduce an assets capacity to 0% in the AEF society. This functionality is available on each of the squadrons, wings, and beddown locations.

1. Select the Assets/Allocations tab (the only tab on wing and beddown window).

2. Double click on one of the assets in the table. This will bring up the Asset Allocation Details dialog. This dialog lists all of the specific assets of the type selected and indicates which of the assets have been allocated.

3. Click on one of the Item Number cells. This will result in a confirmation dialog being displayed.

4. If you want to remove the selected asset (i.e., set it's capacity to 0) click on the "Remove Asset" button of the Delete Record Dialog, otherwise select "Cancel".

The AEF society development team teamed with another company called 21st Century Technologies to develop a fuel assessor plugin. The primary functionality of this plugin was to monitor the planned daily delivery of fuel needed to support the desired sortie rate versus the actual consumption rate. The fuel assessor plugin utilized mathematical models to project future fuel usage based on historical consumption rates and future desired sortie rates. If the plugin determined a potential shortfall, an additional task requesting fuel would be generated. The combination of this fuel assessor plugin and the automatic reordering of fuel was another dynamic replanning aspect added to the AEF society under this effort.

The final perturbation supported in the AEF demonstration society is the ability to dynamically change the specific aircraft selected for the AEF deployment. This functionality allows the user to override decisions made by the squadron cluster on which aircraft were selected based on historical maintenance data. If the user changes the aircraft selected for the deployment, the plugins in the squadron will determine if a different amount of support equipment is needed as a result of the changes. If the plugin determines a different amount of support equipment is needed it will replan accordingly. This replanning will consist of rescinding excess support equipment and allocating additional support equipment. Details on how the user selects different aircraft is described in the user interface section of this document.

DLA Integration and Interaction

Another team working on the overall ALP project developed a set of clusters and plugins representing organizations involved supplying sustainment assets and maintaining inventories. These clusters and plugins were referred to as the DLA society. The assets modeled in the DLA society included fuel and critical spare parts. During this effort, the AEF society was enhanced to take advantage of this functionality in the DLA society.

Recall from the High Level Task Flow section, each of the squadrons involved in the deploying AEFs sends a task to the beddown location. This task identifies the type and quantity of aircraft that will be arriving. Additional information is contained in the tasks that identify the anticipated OPTEMPO for the aircraft arriving. Upon receiving this task, the beddown location generates tasks to the DLA cluster community so it can establish support for fuel and critical spare parts for the aircraft. ALPINE provided the team access to a database which contained data on the fuel burn rate and statistical failure rates of critical spare parts for each type of Mission Design Series (MDS). This database was accessed when creating tasks going to the DLA cluster community.

More detailed interaction with the DLA cluster community is anticipated during the next phase of the AEF cluster community.

TRANSCOM Integration and Interaction

The TRANSCOM cluster society was the other society that was utilized by the AEF society. Air, ground, and sea transportation logic is represented in the TRANSCOM cluster community. The society is designed to respond to transport tasks that have associated with them a collection of assets that need to be transported. Each of the assets attached to the task has properties defining their physical characteristics to include mass, length, width, and height. Upon receiving a transport task, the TRANSCOM clusters plan for the transportation of the assets. The interaction between the AEF society and TRANSCOM society focused on the air transportation functionality. A unique requirement that the team requested of the TRANSCOM society developers was for a special type of task to be sent back to the requesting cluster. This task, referred to as a PFT (Prepare For Transport) task, identified a schedule of when the assets would need to be ready for transport. Additionally, the PFT task identified the type of cargo plane, its itinerary, and the expected manifest for the aircraft. This task was utilized in the Chalk View user interface, which is discussed in the AEF User Interface section.

In addition to physical assets needed to be transported, the AEF society also made requests of the TRANSCOM cluster community to provide transport for personnel supporting the deployment. Upon receiving a request to move personnel assets, the TRANSCOM cluster society forwarded these tasks on to the Virtual Airlines cluster society that was responsible for handling requests for transporting personnel. As with the physical asset transport requests, the Virtual Airline cluster community responded back to the requesting organization with a PFT task indicating when and what type of aircraft would satisfy the request.

AEF Society User Interfaces

This section provides descriptions of the user interface components developed for the AFRL/HESR demonstration software. In order to provide a visual representation of what was happening in the demonstration, as well as provide the user with the ability to interact with the demonstration, user interfaces were developed for the majority of the Air Force clusters. As described earlier, UI plugins were developed for each of the clusters, which provided the capability to access elements of the cluster's LogPlan for display purposes. Additionally, the Chalk View UI was constructed utilizing the PSP mechanism described above.

The following sections provide detailed descriptions and a series of screen captures for the user interface components developed. All of the user interface components were developed utilizing the Java Foundation Classes Swing (JFC) library classes available in the JDK. One other product called JChart, developed by XRTGraph, was utilized to develop the charting capabilities.

JFACC View

As described in the section titled Standalone AEF Cluster Society, the JFACC cluster is command cluster that creates the initial task to start the society running. A simple user interface for the JFACC cluster was created that provides the functionality to get the cluster society started. As can be seen in Figure 6, the JFACC view consists of several buttons. The first button, "Propagate OPLAN", when selected causes the OPLAN to be propagated through the cluster society. This needs to be accomplished prior to the first task being generated. The second button, "Get Log Support", results in the creation of the root level task that starts the AEF society planning. The next button, "Insert AEF9", results in a change being published to the OPLAN that indicates the need for the deployment of an additional AEF (additional to the two generated to support the OPLAN). The "Test Transport" button is used to send a dummy task to the TRANSCOM cluster, this was used to aid in integration testing. The "Pause" button results in a task being published throughout the society directing the clusters to stop processing tasks. This allows the user of the demonstration to observe the state of the system at various stages of plan development. The "Scenario Clock" button results in another window being displayed as shown in Figure 7. This window allows the user to configure the simulation clock available in ALP. From this window the user can select the ratio between real time and scenario time. The Test LeadWingUtcTailor button was added to test the user based tailoring functionality.

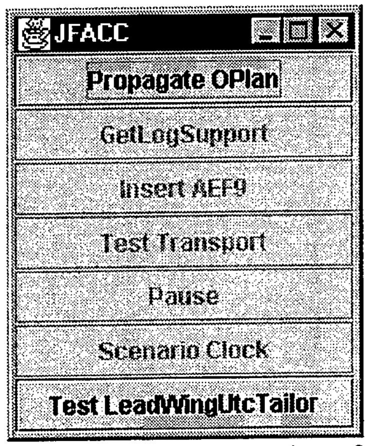


Figure 6: JFACC User Interface

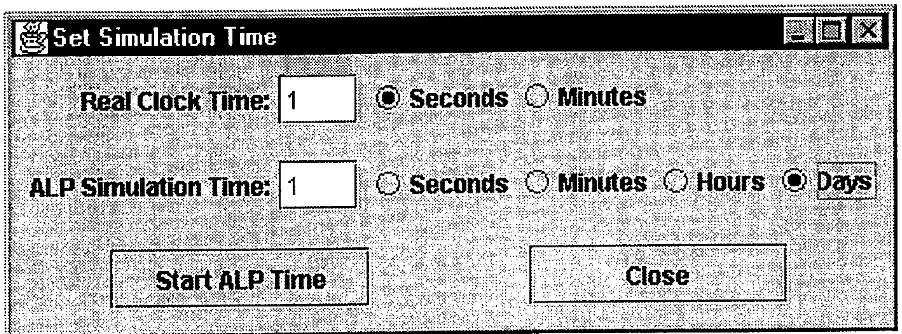


Figure 7: Set Simulation Time User Interface

Squadron User Interface

The squadron user interface is the most complex of all the user interfaces constructed. It provides the user with a great deal of detail on the squadron's fleet of aircraft, equipment, and personnel. In addition to providing the user with status information, the squadron user interface allows the user to interact with the plan being developed. This is accomplished by allowing the user to override the aircraft selected by the ALP system. Details on how this is accomplished will be described later. A single plugin was created which interacts with the log plan to extract the needed data for the squadron user interface. This same plugin can be attached to any of the squadrons in the AEF cluster society in order to create the squadron user interface.

Squadron Main Window View

Figure 8 is the main window that appears for the squadron user interface. Three main sections exist within this window. On the left side, is a series of four tabbed panes that provide detailed information on the aircraft, non-aircraft assets, and personnel of the squadron. The upper right section identifies information about the operation for which the plan is being generated. Finally, the lower right section contains two graphs that provide detail on the squadron's personnel and equipment deployment requirements. The following sections provide more detailed descriptions and screen captures about each of these three sections. The screen captures used in this section are based on 421st Fighter Squadron, but the same user interfaces are available for all the squadrons in the AEF cluster society. In addition to the squadron view discussed in the sections below, a PSP version of this interface was also created. It only provides the icon view of the aircraft and it has the same right hand side view. The primary advantage of this view is that it can be viewed on any machine that has a network connection to the squadron cluster.

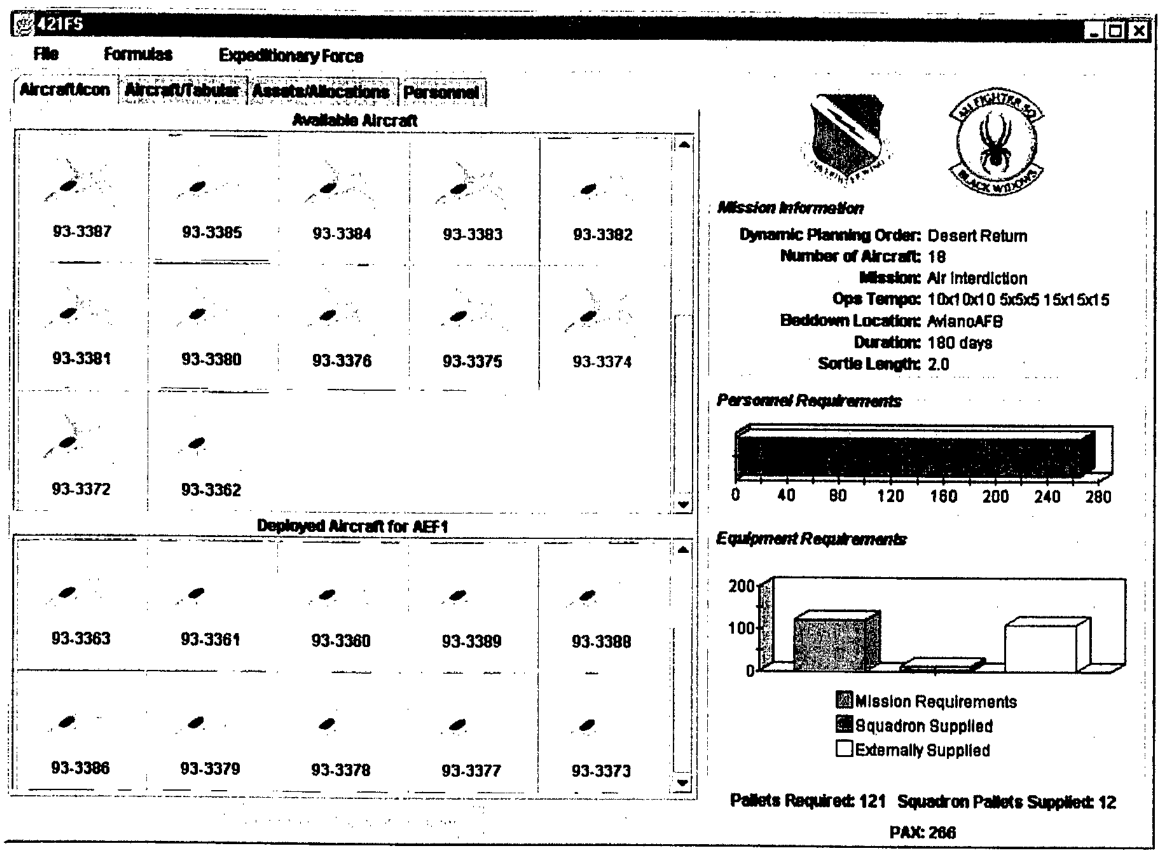


Figure 8: Squadron User Interface

Aircraft Icon View

The aircraft icon view, as shown in Figure 9, provides an interactive view of squadron's aircraft assets. Each aircraft is represented as a square button containing a graphical image of the aircraft as well as the aircraft's tail number. The "deployment status" of each aircraft is also depicted by the background color of the button. The background is set to green to indicate a "good status," yellow to indicate a "fair status," or red to indicate a "poor status." This status is based on calculations done on historical data for each aircraft. The status is not intended to indicate the current mission capability of the aircraft, but rather it is intended to show the current deployability of the aircraft.

This view is separated into an upper and lower section titled "Available Aircraft" and "Deployed Aircraft," respectively. Aircraft appearing in the "Available Aircraft" section are available for a deployment, whereas aircraft appearing in the "Deployed Aircraft" section have been selected for the mission being planned. As mentioned earlier, the user can move aircraft from the deployment section to the available or from the available section to the deployed section simply by dragging and dropping the aircraft icons. This functionality allows the user to override the aircraft selected by the computer for the deployment. If the user has overridden the deployed aircraft selected by the system, the "Confirm Deployment" button will become active in order for the user changes to affect the ALP log plan. Select the "Confirm Deployment" button to begin committing changes. Once selected, the user will be presented with a dialog box detailing the changes (Figure 10). Once the user selects the "Commit Deployment Changes" button on this dialog, the log plan will be adjusted to reflect the user changes.

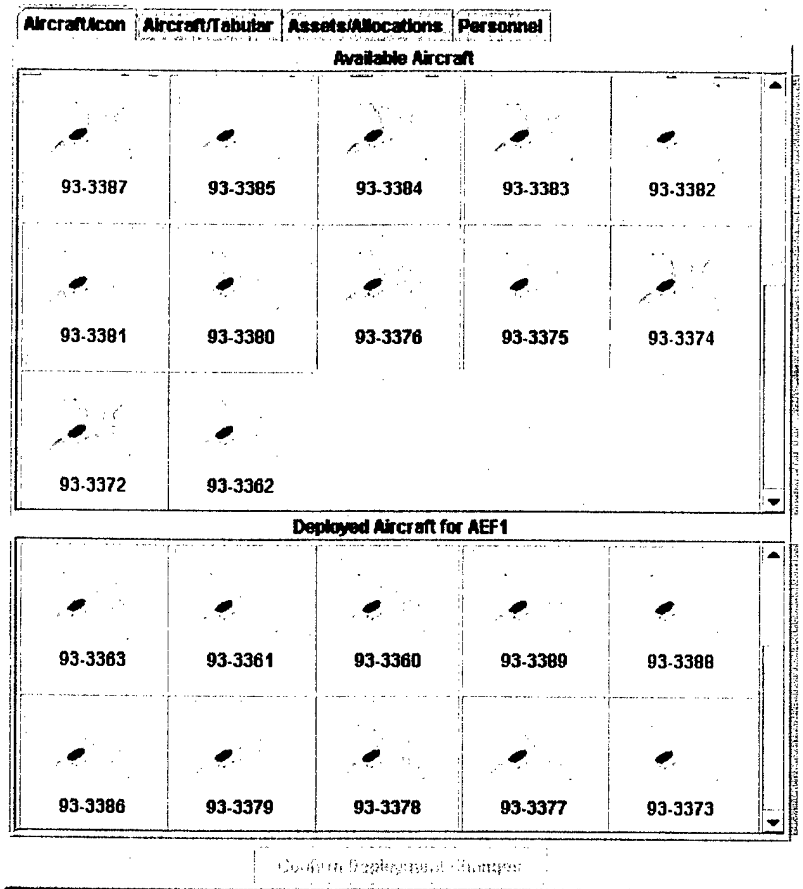


Figure 9: Aircraft/Icon Tab User Interface

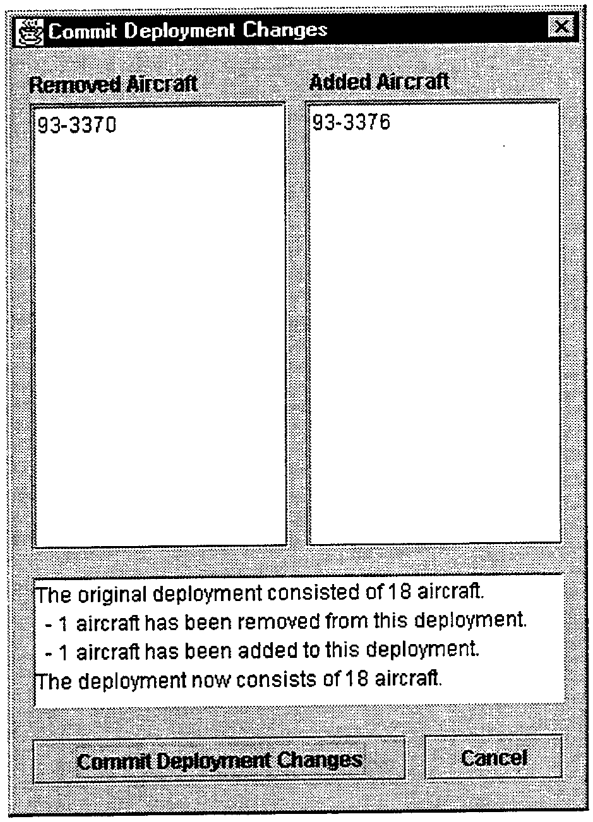


Figure 10: Commit Deployment Changes User Interface

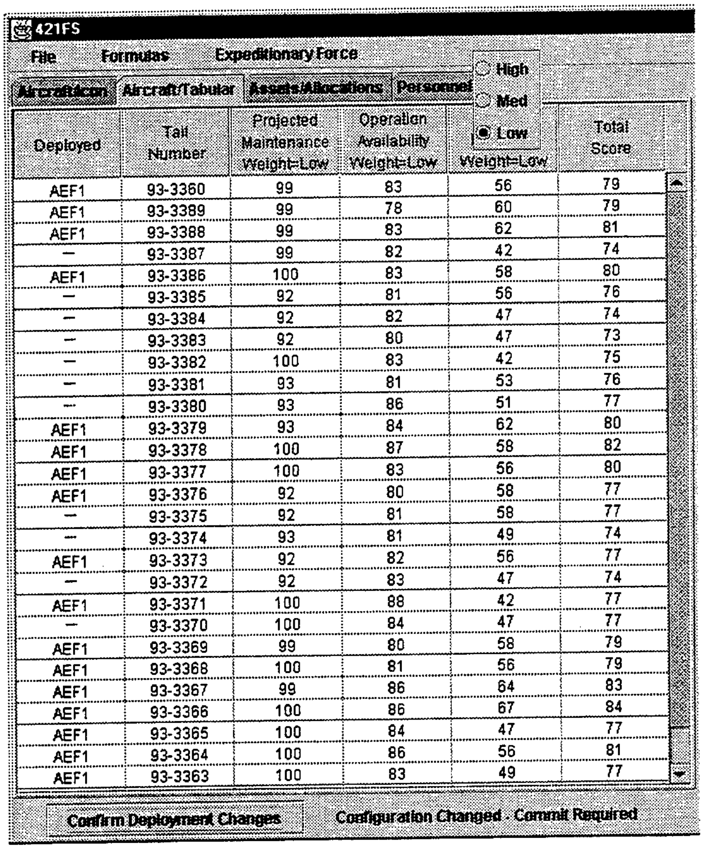
Aircraft Tabular View

The Aircraft Tabular View, as shown in Figure 11, provides another view of the squadron's aircraft assets. This view presents a more detailed view of the aircraft than is available from the icon view. As described in the previous section, each aircraft has a status. For this demonstration, three factors were used to determine an aircraft's status. These factors included a projected maintenance score, operational availability score, and mission reliability score for each aircraft. Historical flight and maintenance data was obtained for the different MDS. This data was then applied to formulas developed by the team yielding the three different scores.

The three factors were selected to address both maintenance and operations concerns. When selecting an aircraft for deployment, a maintainer's primary concern is the status of the aircraft's projected maintenance schedule. For example, if an aircraft has an engine replacement scheduled after an additional 30 flight hours, the maintainer might suggest not taking this aircraft for a deployment because of the additional support equipment requirements and time needed for the engine replacement. The other two factors address operational concerns. These factors are the mission reliability and operational availability factors. The mission reliability provides an accumulative value based on a calculation of the aircraft's recent return codes. If the aircraft had anything other than a "Code 1" (no problem reported) return code the score would be lowered. The operational availability score is calculated for each month based on the total fully mission capable hours of the aircraft.

The Aircraft Tabular view consists of a table with six columns. The first column identifies whether or not a particular aircraft is selected for a particular deployment (indicated with the AEF name) or if the aircraft is available (indicated with a dash "-"). The next column lists the aircrafts' tail numbers. The remaining columns provide the values of the aircraft's projected maintenance, operational availability, mission reliability, and total score.

Another feature available on this view is the ability to change the relative weights for the projected maintenance, operational availability, and mission reliability scores. By selecting any of these column headings with the right mouse button, the user can select a high, medium, or low weight for the column. This feature allows the user to place relative higher or lower precedence on each of the three scores depending on the deployment situation. If the user knows a very short deployment is expected, the projected maintenance score may be given a lower weight. If the weight of a particular category is changed, the system will recalculate the score for that category and update the overall score for the aircraft. This change will then be reflected in the aircraft selected by the ALP system for the plan.



421FS

File Formulas Expeditionary Force

Aircraft/Icon Aircraft/Tabular Assets/Allocations Personnel

High Med Low

Deployed	Tail Number	Projected Maintenance Weight=Low	Operation Availability Weight=Low	Weight=Low	Total Score
AEF1	93-3360	99	83	56	79
AEF1	93-3389	99	78	60	79
AEF1	93-3388	99	83	62	81
—	93-3387	99	82	42	74
AEF1	93-3386	100	83	58	80
—	93-3385	92	81	56	76
—	93-3384	92	82	47	74
—	93-3383	92	80	47	73
—	93-3382	100	83	42	75
—	93-3381	93	81	53	76
—	93-3380	93	86	51	77
AEF1	93-3379	93	84	62	80
AEF1	93-3378	100	87	58	82
AEF1	93-3377	100	83	56	80
AEF1	93-3376	92	80	58	77
—	93-3375	92	81	58	77
—	93-3374	93	81	49	74
AEF1	93-3373	92	82	56	77
—	93-3372	92	83	47	74
AEF1	93-3371	100	88	42	77
—	93-3370	100	84	47	77
AEF1	93-3369	99	80	58	79
AEF1	93-3368	100	81	56	79
AEF1	93-3367	99	86	64	83
AEF1	93-3366	100	86	67	84
AEF1	93-3365	100	84	47	77
AEF1	93-3364	100	86	56	81
AEF1	93-3363	100	83	49	77

Confirm Deployment Changes Configuration Changed - Commit Required

Figure 11: Aircraft/Tabular User Interface

Additional detail of how the score was calculated can be obtained by double clicking on any of the rows in the Aircraft/Tabular view. Figure 12 shows the dialog window that results from selecting a particular aircraft. The figure shows each of the three tabs selected on the dialog window. This dialog provides details on the maintenance data used to calculate its score. A series of three tabs are presented which allow the user to analyze the flight history, operational availability, and projected maintenance data available for the particular aircraft. Additionally, the upper section of the dialog summarizes the raw score for each category and the weight applied to each of these categories.

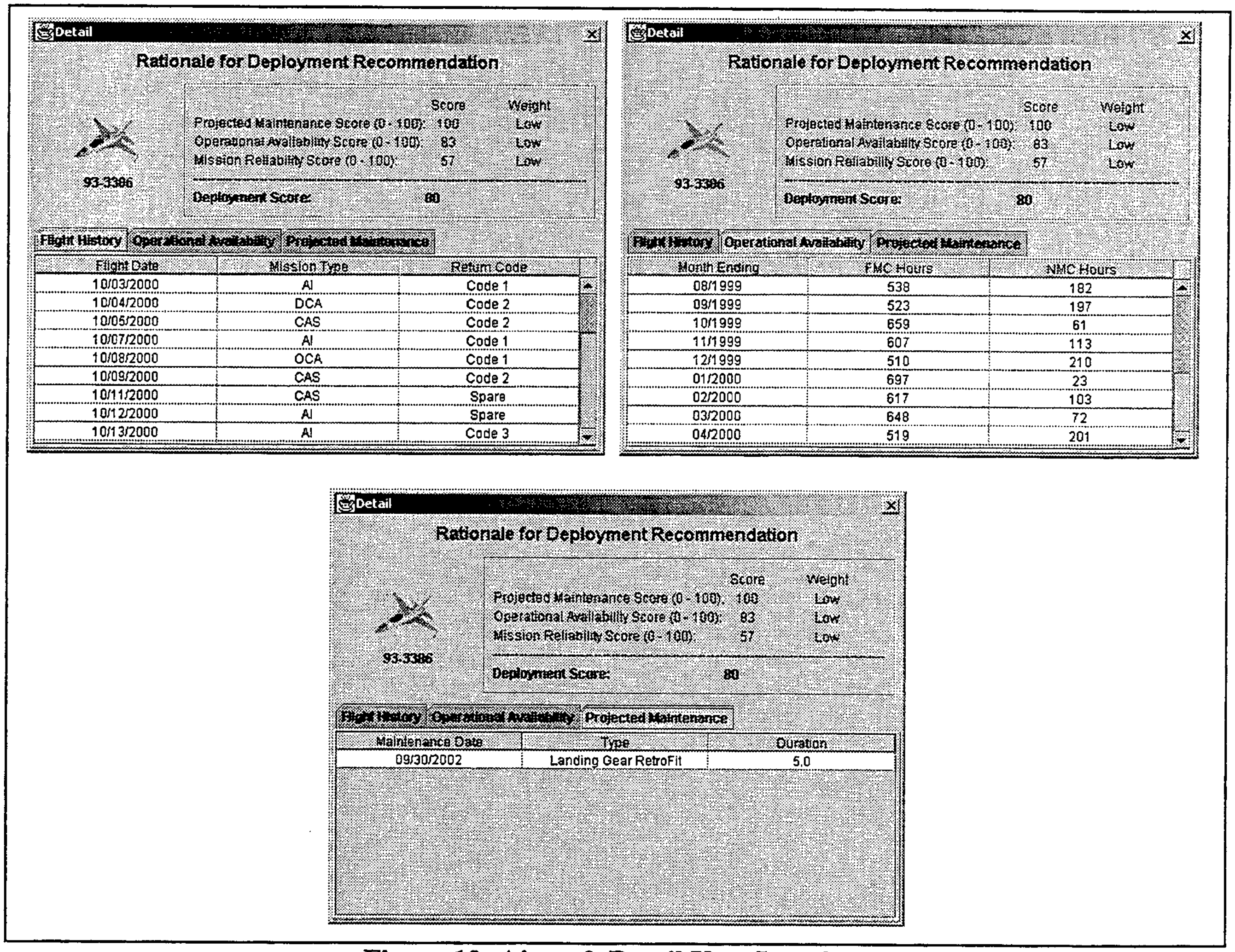
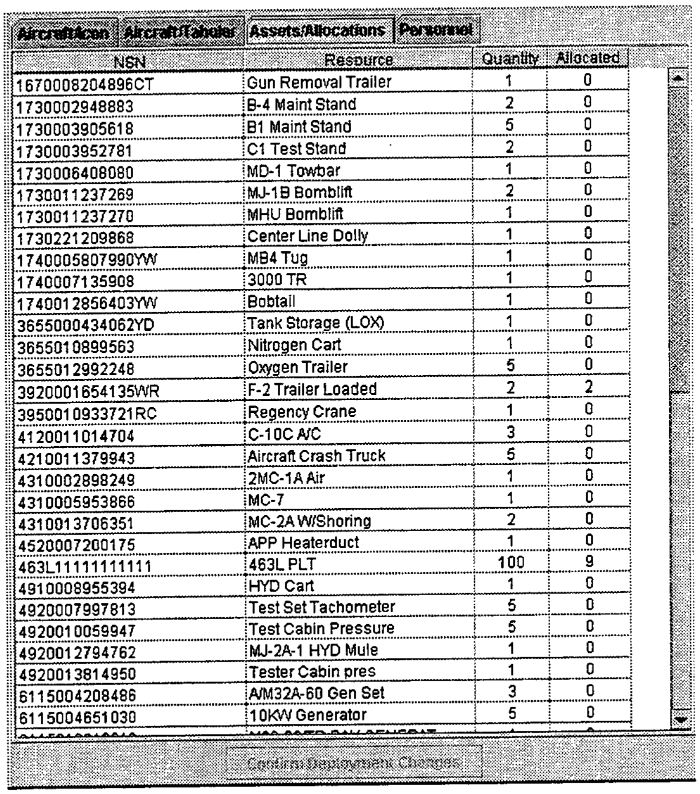


Figure 12: Aircraft Detail User Interface

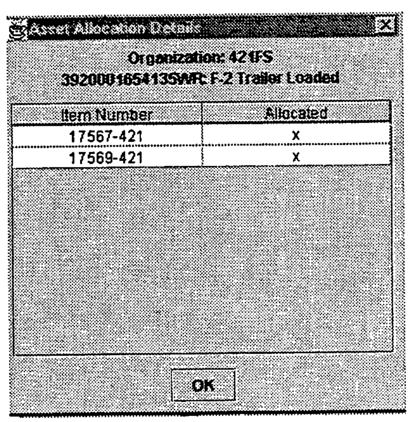
Assets Allocations View

The third tab on the squadron user interface, as shown in Figure 13, is titled "Assets/Allocations" view. This view provides details on the non-aircraft assets available at the squadron. As can be seen from Figure 13, this view consists of a table with four columns. The first column named "NSN" identifies the National Stock Number of the resource. The next column named "Resource" identifies the human readable name or nomenclature of the resource. The third column named "Quantity" indicates the total quantity of the particular asset. The final column, titled "Allocated" lists the quantity of a particular resource the fighter wing has allocated. Each of the rows in the assets allocation view can be selected by executing a double click action. This will result in a dialog being displayed that provides additional information on the particular asset. The dialog consists of a table with two columns, where the first column indicates the serial number of the item and the second column indicates if the particular item has been selected to support the deployment. An example of the detail available from this dialog can been seen in Figure 14.



NSN	Resource	Quantity	Allocated
1670008204896CT	Gun Removal Trailer	1	0
1730002948883	B-4 Maint Stand	2	0
1730003905618	B1 Maint Stand	5	0
1730003952781	C1 Test Stand	2	0
1730006408080	MD-1 Towbar	1	0
1730011237269	MJ-1B Bomblift	2	0
1730011237270	MHU Bomblift	1	0
1730221209868	Center Line Dolly	1	0
1740005807990YW	MB4 Tug	1	0
1740007135908	3000 TR	1	0
1740012856403YW	Bobtail	1	0
3655000434062YD	Tank Storage (LOX)	1	0
3655010899563	Nitrogen Cart	1	0
3655012992248	Oxygen Trailer	5	0
3920001654135WR	F-2 Trailer Loaded	2	2
3950010933721RC	Regency Crane	1	0
4120011014704	C-10C A/C	3	0
4210011379943	Aircraft Crash Truck	5	0
4310002898249	2MC-1A Air	1	0
4310005953866	MC-7	1	0
4310013706351	MC-2A W/Shoring	2	0
4520007200175	APP Heaterduct	1	0
463L11111111111	463L PLT	100	9
4910008955394	HYD Cart	1	0
4920007997813	Test Set Tachometer	5	0
4920010059947	Test Cabin Pressure	5	0
4920012794762	MJ-2A-1 HYD Mule	1	0
4920013814950	Tester Cabin pres	1	0
6115004208486	A/M32A-60 Gen Set	3	0
6115004651030	10KW Generator	5	0

Figure 13: Assets/Allocations Tab User Interface

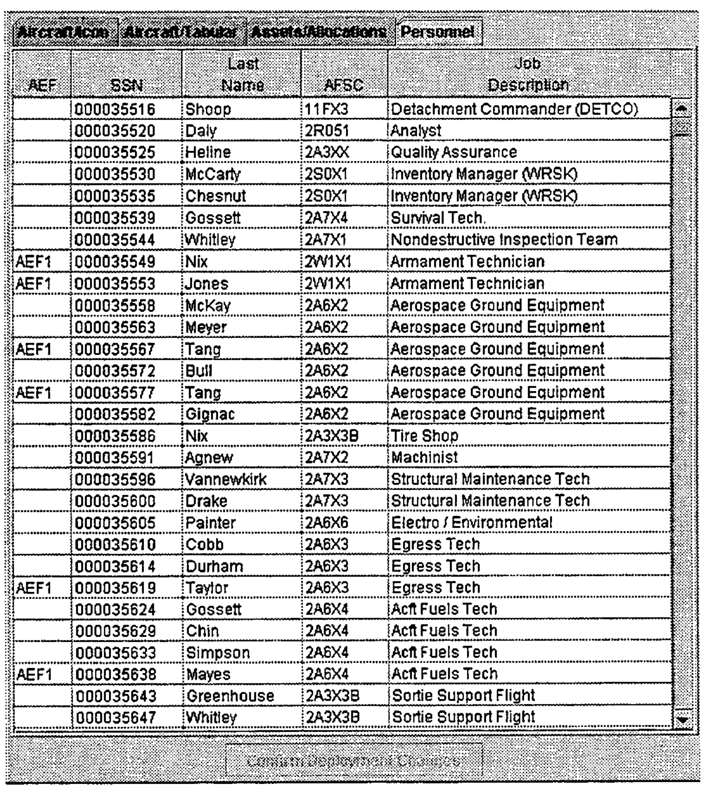


Item Number	Allocated
17567-421	x
17569-421	x

Figure 14: Asset Allocation Detail User Interface

Personnel View

The final tab on the squadron user interface, as shown in Figure 15, is titled "Personnel" view. This view provides detailed information for all of the personnel assigned to the squadron. The view consists of a table with five columns. The first column indicates whether the individual is selected for the deployment being planned. If the person is selected, the particular AEF that the person is supporting will be indicated in this column. If the column contains nothing, the individual is currently not selected for the deployment. The second column indicates the social security number of the individual. The third column indicates the individual's last name. The final two columns identify the individuals Air Force Specialty Code (AFSC) and the individual's job description. Additional information about each individual can be viewed by executing a double click action on any of the rows in the table. This will result in a dialog being displayed like the one shown in Figure 16. This detail dialog provides information on the selected individual's deployment history, training history, and immunizations. For this effort, only notional data was created for the immunization records. Note, that the personnel data utilized in this demonstration is purely fictitious and the social security numbers and names were programmatically generated.



AEF	SSN	Last Name	AFSC	Job Description
	000035516	Shoop	11FX3	Detachment Commander (DETCO)
	000035520	Daly	2R051	Analyst
	000035525	Heline	2A3XX	Quality Assurance
	000035530	McCarty	2S0X1	Inventory Manager (WRSK)
	000035535	Chesnut	2S0X1	Inventory Manager (WRSK)
	000035539	Gossett	2A7X4	Survival Tech.
	000035544	Whitley	2A7X1	Nondestructive Inspection Team
AEF1	000035549	Nix	2W1X1	Armament Technician
AEF1	000035553	Jones	2W1X1	Armament Technician
	000035558	McKay	2A6X2	Aerospace Ground Equipment
	000035563	Meyer	2A6X2	Aerospace Ground Equipment
AEF1	000035567	Tang	2A6X2	Aerospace Ground Equipment
	000035572	Bull	2A6X2	Aerospace Ground Equipment
AEF1	000035577	Tang	2A6X2	Aerospace Ground Equipment
	000035582	Gignac	2A6X2	Aerospace Ground Equipment
	000035586	Nix	2A3X3B	Tire Shop
	000035591	Agnew	2A7X2	Machinist
	000035596	Vannewkirk	2A7X3	Structural Maintenance Tech
	000035600	Drake	2A7X3	Structural Maintenance Tech
	000035605	Painter	2A6X6	Electro / Environmental
	000035610	Cobb	2A6X3	Egress Tech
	000035614	Durham	2A6X3	Egress Tech
AEF1	000035619	Taylor	2A6X3	Egress Tech
	000035624	Gossett	2A6X4	Acft Fuels Tech
	000035629	Chin	2A6X4	Acft Fuels Tech
	000035633	Simpson	2A6X4	Acft Fuels Tech
AEF1	000035638	Mayes	2A6X4	Acft Fuels Tech
	000035643	Greenhouse	2A3X3B	Sortie Support Flight
	000035647	Whitley	2A3X3B	Sortie Support Flight

Figure 15: Personnel Tab User Interface

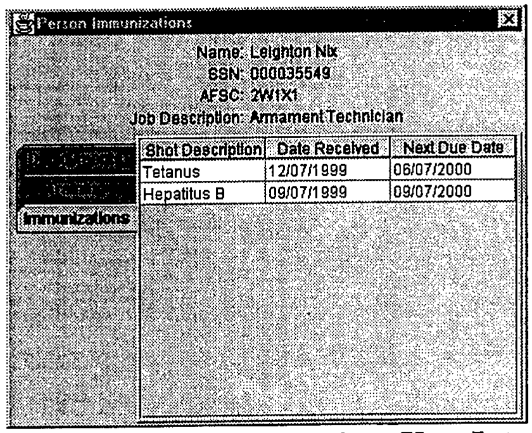


Figure 16: Person Immunizations User Interface

Mission Information

The right hand side of the squadron view main window contains information about the mission for which the plan is being generated. At the top portion of this window, the crest of the wing to which the squadron belongs is displayed along with the crest of the squadron. The mission information contains the Dynamic Planning Order name and the mission name. The number of aircraft to be deployed and the planned OPTEMPO, including average planned sortie length, is also presented. Finally, the location of where the squadron is going to bed down and the length of deployment are available in this view as can be seen in Figure 17.

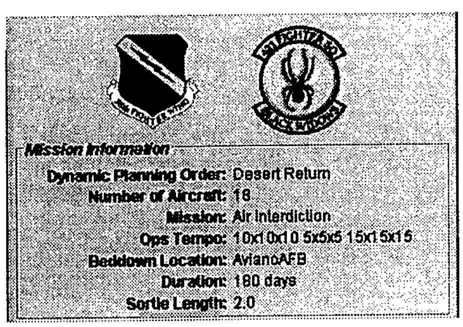


Figure 17: Squadron Information

Personnel and Equipment Requirements

The final pieces of information available on the squadron view's main window are detail on the personnel and equipment requirements for the squadron (Figure 18). The personnel requirements graph indicates the total number of personnel that are to be deployed to support the deploying squadron. This includes base, indirect, and direct combat personnel. The equipment requirements graph consists of three bars representing the quantities of 463L pallet positions of equipment assets. The first column indicates the mission requirements, which is the total number of pallet positions need by the squadron to support the number of deploying aircraft and the planned sortie rate and sortie duration. This quantity was obtained from Unit Type Codes (UTC) supplied by AFRL/HESR. The second column indicates the amount of pallet positions the squadron is going to organically supply. This amount will typically be less than the total required as a result of being able to fulfill the squadron's equipment requirements by in-theater or near-theater sources. The final column indicates the difference between the required and supplied pallet positions externally supplied.

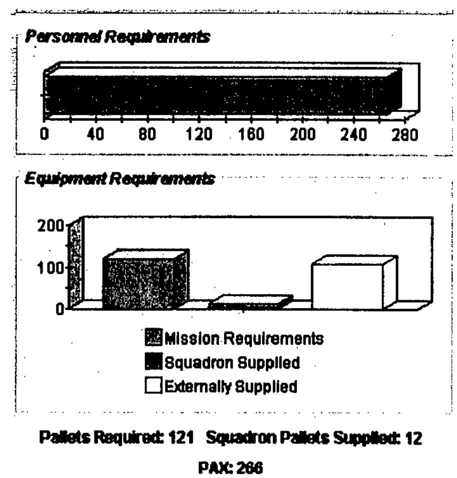
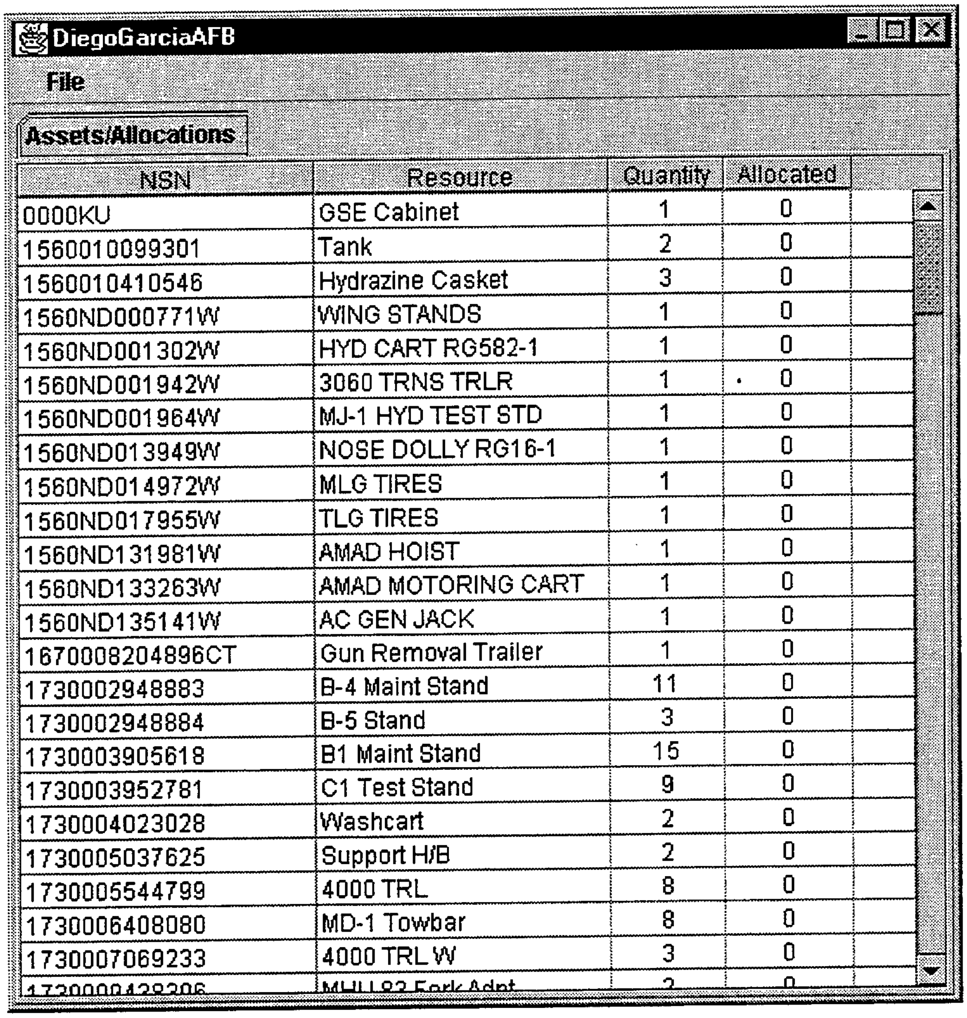


Figure 18: Squadron Graphs

Wing and Beddown Views

A simple user interface exists for each of the wing clusters and beddown clusters in the AEF society. Like the squadron user interface, the view available for the wings and beddown clusters has a plugin that retrieves the information from the log plan and provides it to the view. The same software is used for each of the wings and beddown clusters; it is simply a matter of adding the plugin to the cluster to make this view appear. The wing and beddown view provides the user with a table that is identical to the assets tab in the squadron view. This table consists of four columns which are titled: NSN, Resource, Quantity, and Allocated. This view provides the user visibility into the assets and their allocation status.



DiegoGarciaAFB

File

Assets/Allocations

NSN	Resource	Quantity	Allocated
0000KU	GSE Cabinet	1	0
1560010099301	Tank	2	0
1560010410546	Hydrazine Casket	3	0
1560ND000771W	WING STANDS	1	0
1560ND001302W	HYD CART RG582-1	1	0
1560ND001942W	3060 TRNS TRLR	1	0
1560ND001964W	MJ-1 HYD TEST STD	1	0
1560ND013949W	NOSE DOLLY RG16-1	1	0
1560ND014972W	MLG TIRES	1	0
1560ND017955W	TLG TIRES	1	0
1560ND131981W	AMAD HOIST	1	0
1560ND133263W	AMAD MOTORING CART	1	0
1560ND135141W	AC GEN JACK	1	0
1670008204896CT	Gun Removal Trailer	1	0
1730002948883	B-4 Maint Stand	11	0
1730002948884	B-5 Stand	3	0
1730003905618	B1 Maint Stand	15	0
1730003952781	C1 Test Stand	9	0
1730004023028	Washcart	2	0
1730005037625	Support H/B	2	0
1730005544799	4000 TRL	8	0
1730006408080	MD-1 Towbar	8	0
1730007069233	4000 TRL W	3	0

Figure 19: Wing/Beddown User Interface

Rule Based Tailoring View

The rule based tailoring view provides the user the ability to construct tailoring rules which are utilized when consolidating the combined UTCs from the deploying squadron for a particular AEF. This view as shown in Figure 20, is accessible from the Wing user interface by selecting the File menu. Building a rule consists of creating a condition and resulting action. The condition specifies what criteria must be met and the action specifies what will be done if the condition tests true. In the current implementation, the rules are applied in a linear fashion; there is currently no way to combine the rules with binary logic (AND, OR, etc). If this were to become a fielded system, the rule based tailoring functionality would need to become more robust and feature rich. The primary design goal for this functionality was to emphasize the user's role in the planning process. This functionality allows the user to override existing decision logic within ALP and create new decision logic.

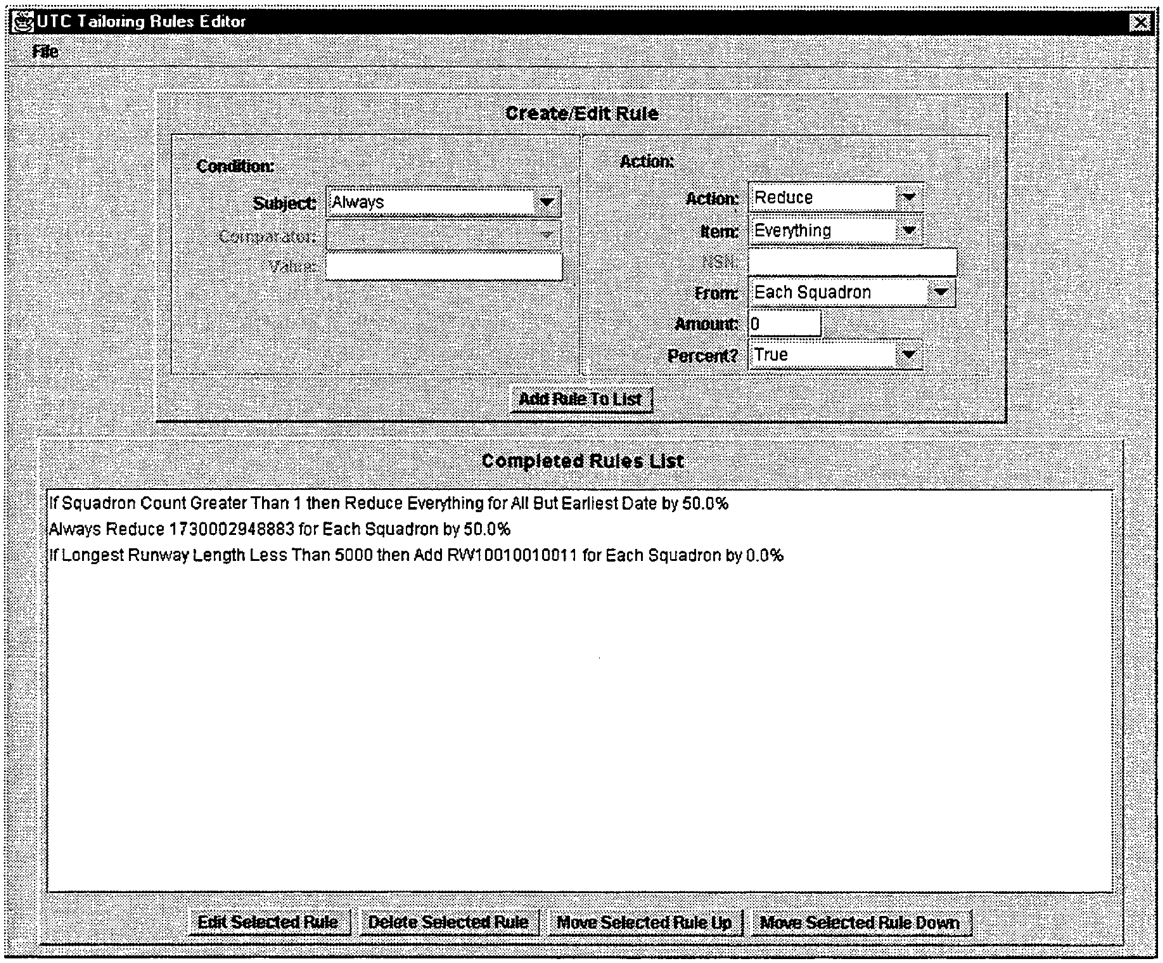


Figure 20: UTC Tailoring Rules Editor User Interface

Log Plan View

Another window that is available from both the squadron main window and the wing beddown window is a view of the LogPlan. Details of the LogPlan were defined in an earlier section titled "LogPlan." This view as shown in Figure 21, provides a tree view of the entire LogPlan for a particular cluster. This view is primarily used by developers as a debugging tool to confirm that the cluster is receiving the correct incoming directives and that it is sending the correct outgoing directives. This view is activated from the File menu on both the squadron main window and the wing/beddown main window. In addition, the log plan view can be attached to any cluster by adding the appropriate plugin to the cluster.

Figure 21: LogPlan User Interface

Chalk View

The last user interface to be discussed in this section is called the Chalk View. This view captures all the details of the transportation requests made by the AEF clusters to the TRANSCOM cluster. This view was constructed using the PSP mechanism discussed in an earlier section. It can be executed on any machine that has a network connection to the AEF cluster society. This view is the only one in the AEF cluster society that provides an aggregate view of details from multiple clusters. This view gathers data from all of the wing clusters in the AEF society to provide the user with an overall view of the status of the transportation of assets and personnel from the AEF organizations to the beddown locations.

As can be seen from Figure 22, this view consists of a tree structure on the left hand side and a Gantt chart on the right hand side. The tree view is organized at the highest level by the individual wings. Each wing can be expanded which will result in a display that lists all of the chalks that are being transported from the particular wing. Each of the chalks can be further expanded to show more detail. This detail includes the MDS and tail number of the cargo plane scheduled to move the chalk, the itinerary, and the manifest of the chalk. The itinerary is broken out into phases consisting of transit, load, and unload. Each of the itinerary phases provides the start and end times as well as the start and end location for the phase. The final piece of information available for the chalk is the manifest, which lists all of the items that make up the chalk. The manifest is organized by NSN and each NSN can be further expanded to reveal the individual item numbers of that class of NSN being transported.

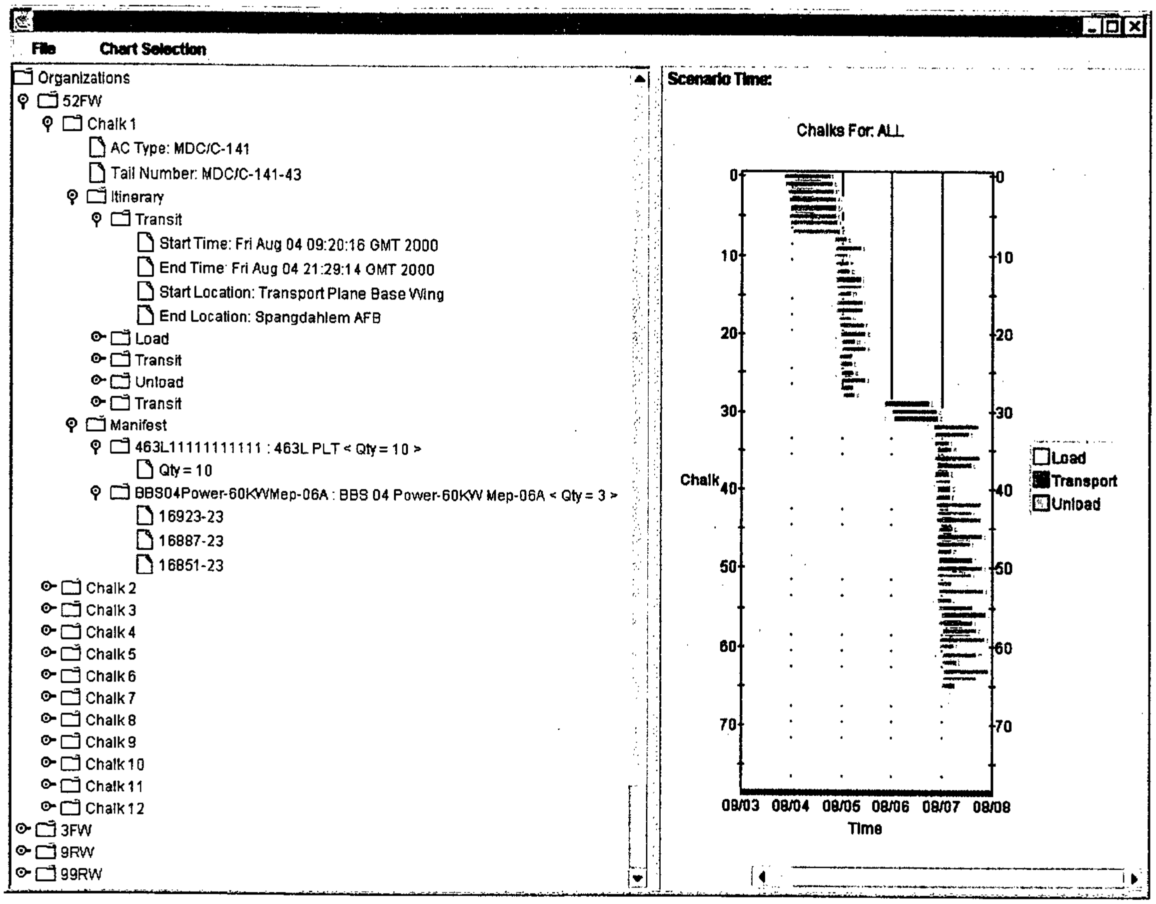


Figure 22: Chalk View User Interface

The right hand side of the chalk view window provides a graphical representation of the detail available in the tree view. This graph consists of a series of horizontal bars, which correspond to the chalks described in the paragraph above. Each of the bars is broken down into three segments: load, transit, and unload. Time is represented along the horizontal axis of the graph and it increases from left to right. Each of the horizontal bars are selectable and when selected will result in a detail dialog being displayed as shown in Figure 23. This dialog provides details on the particular chalk selected. At the top of this detail dialog is the origination and destination of the chalk including the arrival time. Additional information includes the type and tail number of the transporting aircraft. Below this header information is a table that displays the manifest of the chalk. The Chart Selection menu item on the chalk view allows the user to filter information displayed in the chart section of the window. This menu item allows the user to select an individual wing of interest or to select all wings. The "All Wing" view is captured in Figure 22.

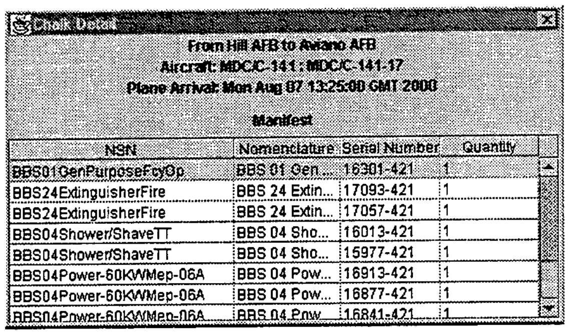


Figure 23: Chalk Detail User Interface

AEF Databases

This section provides details on how all the data used in the AEF society was modeled and populated. The previous version of the AEF cluster society utilized a series of structured flat files to store all the necessary data to drive the AEF society. For this effort, much more information was needed to support both the breadth and the depth of the AEF society. It was decided to utilize a relational database to store the needed information. In an attempt to maintain a portable demonstration, it was decided not to attempt to access any of the production Air Force data systems. Instead, notional representations of these data systems were created. Early in the project an attempt was made to obtain the schemas and data dictionary for the Integrated Maintenance Data System (IMDS). It was not feasible unfortunately, to obtain this information in a timely manner.

In an effort to support distributed network accessible data stores, a Common Object Request Broker Architecture (CORBA) interface was developed. The CORBA interface allows the cluster to access the data through Object Request Brokers (ORBs) using the CORBA architecture. The freely available CORBA implementation available with Sun's JDK 1.2 version was used for the ORB. In addition to the CORBA interface, a more direct Java Database Connectivity (JDBC) mechanism was also developed. Both the CORBA and JDBC connection schemes where abstracted out to an interface layer which makes the client code common regardless if CORBA or JDBC is used as the communication mechanism. A few entries need to be set in one of the configuration files in order to activate the CORBA or the JDBC interface. The AEF plugins require no changes, they simple use the same methods and the abstraction layer determines if CORBA or JDBC should be used. Figure 24 provides a high level overview of the architecture that was created.

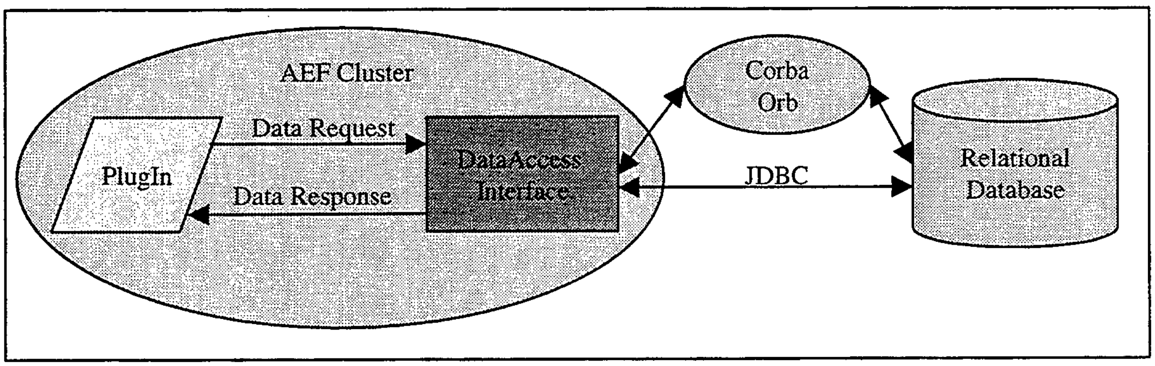


Figure 24: Database Access Mechanism

Database Structure

The following section provides details on the schema that was created for the AEF society and how the database was populated. The database was modeled using the Logic Works ERwin ERX 3.5 data-modeling tool. ERwin has many powerful features that let you design entity relation data models and dimensional models. With ERwin you can create and maintain databases on many different target servers. Within ERwin, there are multiple data modeling notations to choose from. For ALP, the Information Engineering (IE) notation for data modeling is used. The standard IE notation has been modified slightly to take advantage of certain ERwin features.

In IE, a solid line is used to represent an identifying relationship and a dotted line to represent a non-identifying relationship. In IE notation, the child end indicates the cardinality with:

- Crows feet with a cross and a circle, as shown below (Zero, one or more)
- Crows feet with a cross (One or more)
- A cross and a circle (Zero or one)
- A cross (Exactly 1)

An identifying relationship is a relationship between two tables in which an instance of a child table is identified through its association with a parent table, which means the child table is dependent on the parent table for its identity, and cannot exist without it. In an identifying relationship, **one** instance of the parent table is related to **multiple** instances of the child. ERwin draws, in IE notation, an identifying relationship line as a solid line with crow's feet, and the child table box is slightly rounded. An example of how this is represented in ERwin is shown in Figure 25.

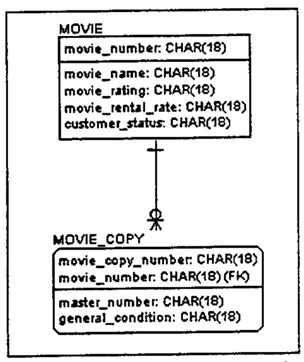


Figure 25: Identifying Relationship Example

A **non-identifying relationship** is a relationship between two tables in which an instance of the child table is not identified through its association with a parent table, which means the child table is not dependent on the parent table for its identify, and can exist without it. In a non-identifying relationship, one instance of the parent table is related to multiple instances of the child.

In an **optional non-identifying relationship**, the columns that are migrated into the non-key area of the child table are not required in the child table. This means that nulls are allowed in the foreign key. ERwin draws an optional non-identifying relationship differently depending on the notation for your diagram. An example of how this is represented in ERwin is shown in Figure 26.

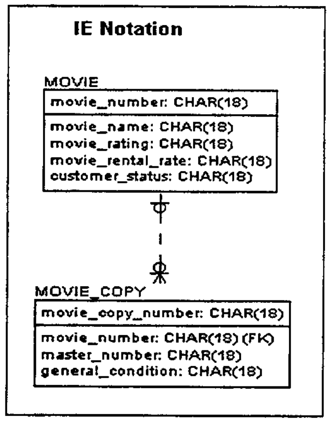


Figure 26: Non-Identifying Relationship Example

Figure 27 illustrates the overall logical model of the ALP AEF database. The ERwin data model includes the ALP database schema, or structure, and all other necessary database objects needed to create the ALP database, whether for Microsoft Access or Oracle. The ERwin tool allowed the team to support both MS Access and Oracle databases seamlessly. The tool generated the schema for both Access and Oracle from a single model. The database was logically divided into seven different databases. Each of these seven logical databases including how they were populated will be further explained in the next sections. A larger printout of the overall AEF database schema is available in Appendix A of this document.

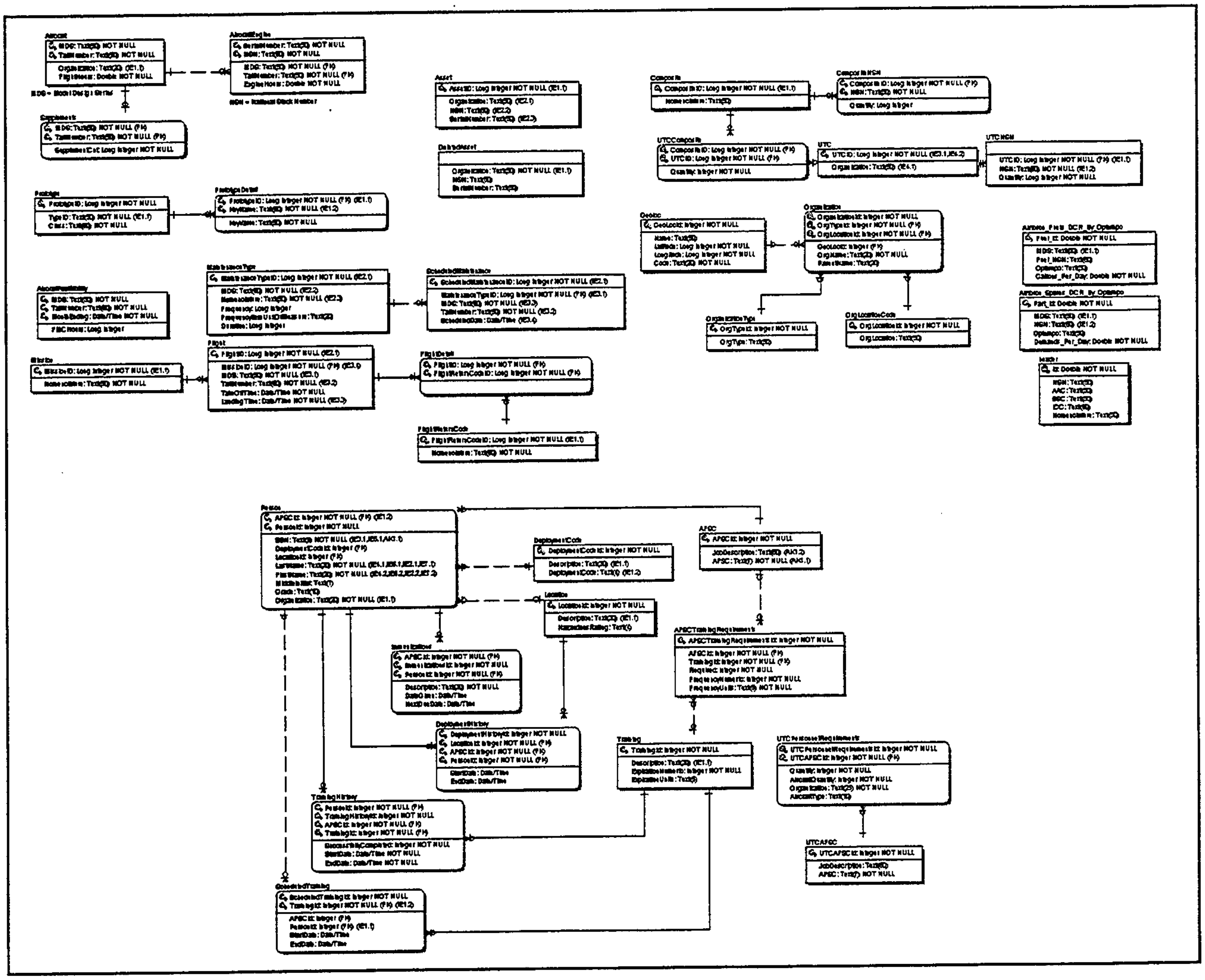


Figure 27: Overall Schema

Maintenance Database

The purpose of the maintenance database was to model scheduled maintenance, reliability, and, flight return codes for each of the aircraft in the AEF society. The entity relationship diagram (ERD) illustrated in Figure 28 shows the table structure that makes up the maintenance database. HESR personnel obtained two months of maintenance data for an F-16 squadron that was used to help create this model. Aircraft reliability refers to how much time during a month a particular aircraft is fully mission capable (FMC).

In order to model aircraft reliability, the AircraftReliability table was created. An AircraftReliablity record was created monthly for each aircraft and the record contained the total FMC hours for the month.

Each time an aircraft executes a flight, any problems experienced are logged and categorized into a particular return code. The Flight, Mission, FlightDetail, and FlightReturnCode tables were constructed to model this aspect. For each flight a flight record was created. This flight record has associated with it the particular type of mission flown. Additionally, the flight record contains 0 or more FlightDetail records which records each of the problem reports logged.

The final aspect modeled in the maintenance database is scheduled maintenance. An example of a scheduled maintenance activity would include things like the ejection seats on F16s must be tested after every 500 flight hours. The MaintenanceType and ScheduledMaintenance tables were constructed to store scheduled maintenance information. For each type of scheduled maintenance, a MaintenanceType record was created. This record identifies the type of aircraft the maintenance applies (e.g. F15) and the amount of time to complete the maintenance. Additionally the frequency at which the maintenance activity takes place was also recorded in the MaintenanceType record. Examples of frequency information are: every 1000 flight hours or every 125 sorties. Records of the Maintenance table were created for each aircraft to identify scheduled maintenance entries for the aircraft. This entry identifies the tail number of the aircraft and the associated MaintenanceType.

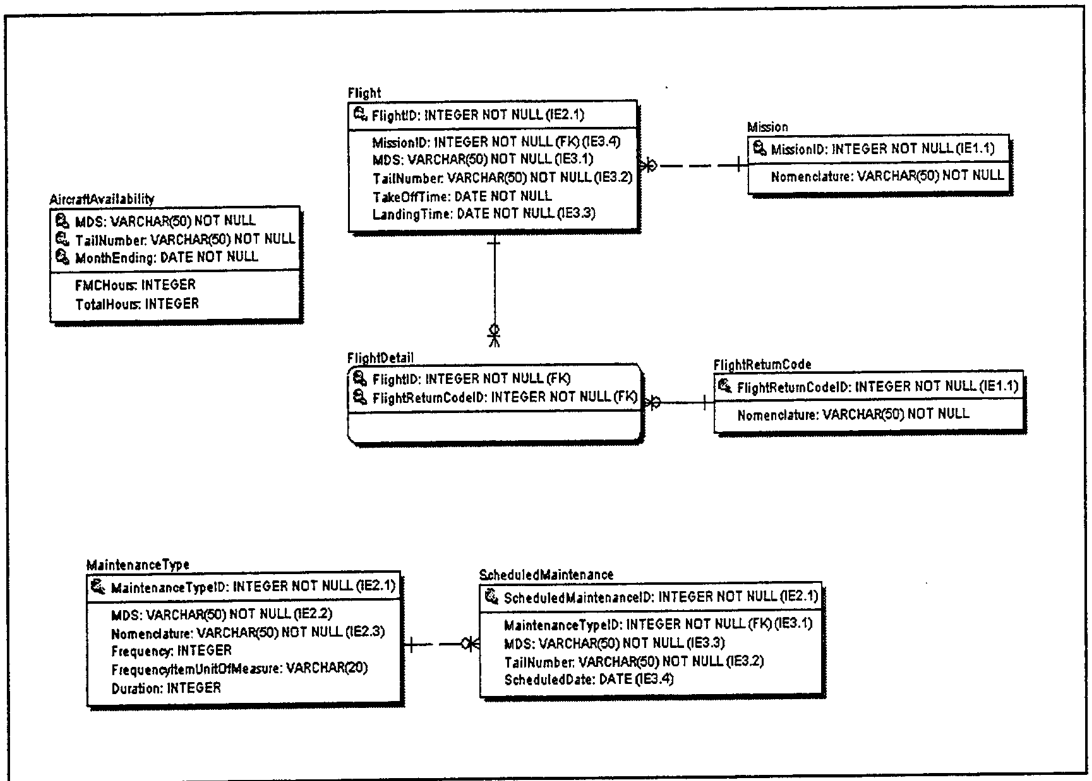


Figure 28: Maintenance Database Schema

The Maintenance database is populated entirely with notional data. The team analyzed the sample maintenance data obtained by AFRL/HESR in order to create the logic to populate the Maintenance database. For each aircraft found in the AircraftAsset database (described below) the populating program creates scheduled maintenance, reliability, and flight detail records.

AircraftAssets Database

The AircraftAssets database (Figure 29) was created to store information on all of the aircraft modeled in the AEF society. The database consists of two tables, which are named Aircraft and AircraftEngine. The Aircraft table stores the name, tail number, and number of flight hours for the aircraft. Additionally, the organization to which the aircraft belongs is also recorded in the Aircraft table. Associated with each aircraft record are one or more AircraftEngine records (depending on the number of engines on a particular aircraft). The AircraftEngine record stores the serial number of the engine and the current number of engine hours.

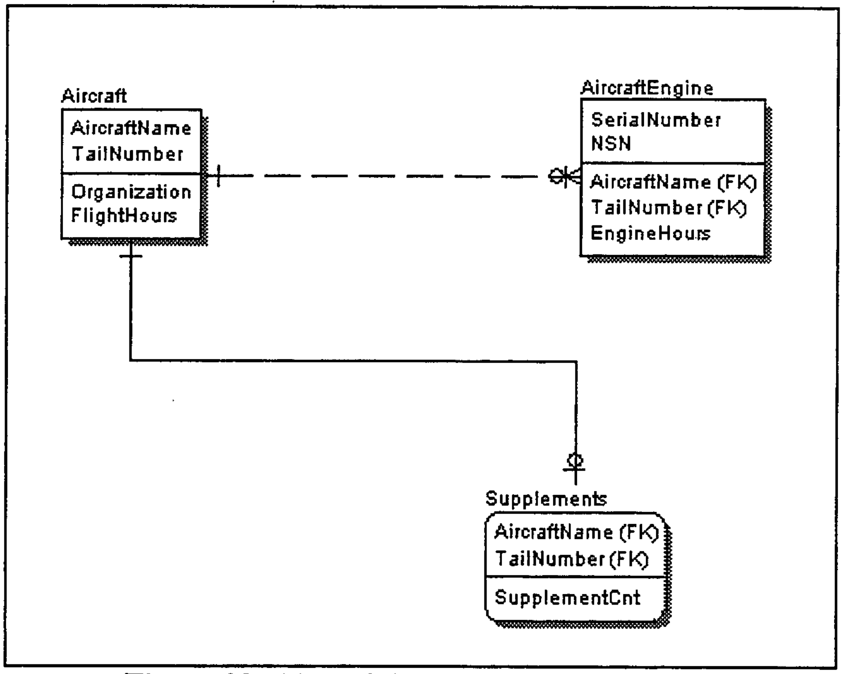


Figure 29: AircraftAssets Database Schema

AFRL/HESR provided data indicating the number and type of each aircraft for each of the squadrons modeled. This data was recorded into comma-delimited files used by the database load program to create the Aircraft and Aircraft engine records.

Assets Database

In order to identify the assets available at each organization in the AEF society, the Assets database was created (Figure 30). This database consists of two tables, which are named Asset and Deleted Asset. The Asset table identifies the National Stock Number (NSN) and serial number for each asset. The organization to which the asset belongs is also recorded in the Asset table. The AssetID field was added as a primary key to improve performance through the use of indexes and adds no additional information.

The Deleted asset table is used during the execution of the AEF demonstration society. During the execution of the AEF society, the user can select individual assets that are to be marked as unavailable. For each of these assets identified, a record of type DeletedAsset is created. Each time the AEF society is run, the DeletedAsset records are purged from the database.

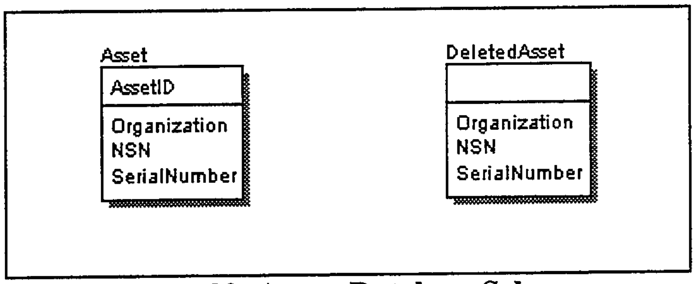


Figure 30: Assets Database Schema

To load the records into the Assets database, the team utilized a series of UTCs obtained by AFRL/HESR. The UTCs were delivered to the team in Microsoft Excel format. These files were then converted to comma separated ASCII text files. A series of programs were then created that parsed the comma-separated files in order to load this data into the different databases including the Asset database.

Prototypes Database

The purpose of the prototypes database is to model characteristics of each unique asset modeled in the Asset database described above. Figure 31 shows the tables that make up the prototypes database. There is a single record in the Prototype table representing each of the assets modeled in the demo system. In this table an automatic id (PrototypeID) is created. The TypeID field stores the name of the NSN associated with the asset. The Class field is used to identify the ALP java class to instantiate when creating this type of asset in ALP. The PrototypeDetail is an association table that allows for the modeling of name value pairs. Examples of what is stored in the Prototype detail include characteristics such as length, width, weight etc. The data used to populate the Prototype database was obtained from UTCs obtained from AFRL/HESR.

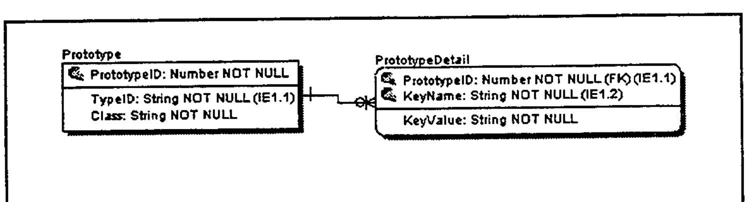


Figure 31: Prototypes Database Schema

Personnel Database

The Personnel database stores information on the different personnel modeled in the AEF society. Figure 32 shows the tables that make up the Personnel database. The primary table in the Personnel database is the Personnel table. The Personnel table stores the social security number, name, grade, and the organization to which the person belongs. The table also has keys, which provide references to other tables in the personnel database. The other tables in the Personnel database allow the deployment history and the current deployment location to be modeled. The ability to model training and immunization information is available, but is currently not being utilized.

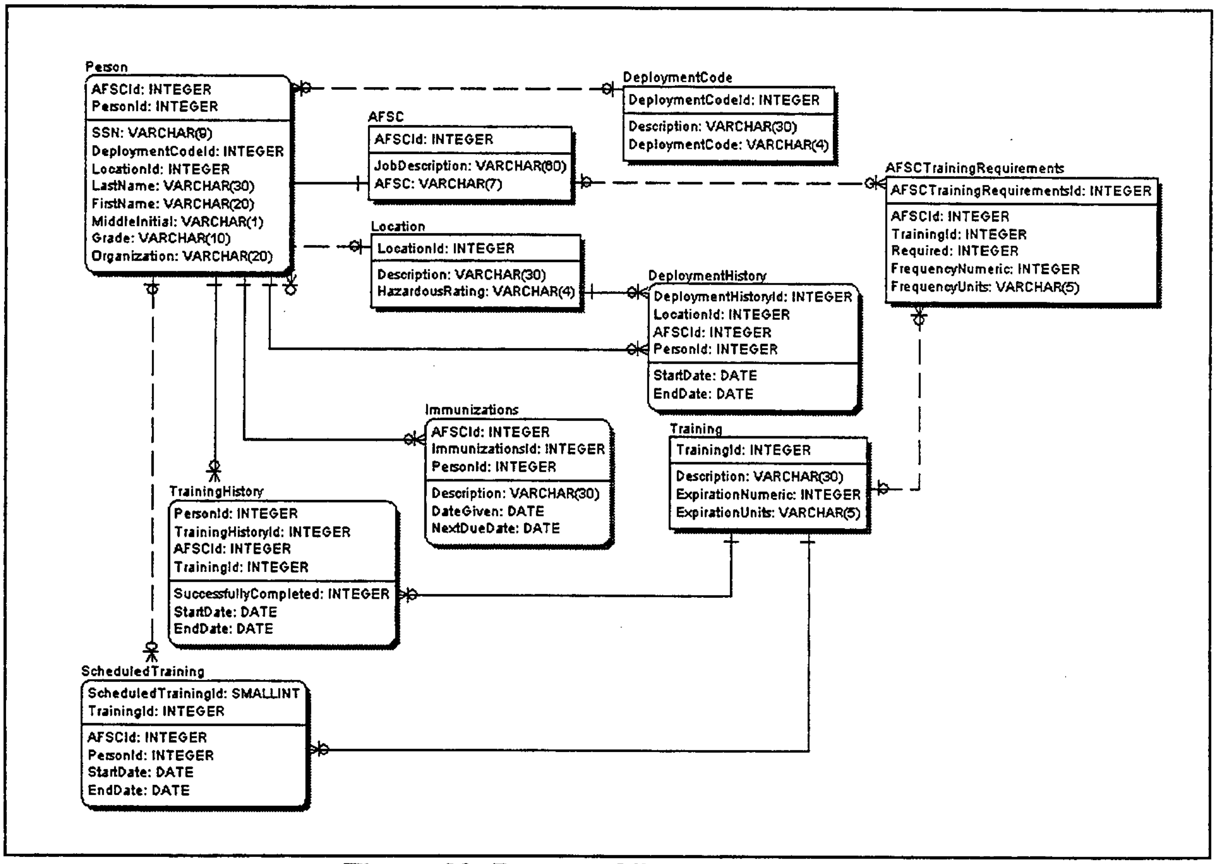


Figure 32: Personnel Database Schema

As with the assets database, a series of personnel UTCs were obtained by AFRL/HESR that provided the needed information to populate the Personnel database. These UTCs were structured such that they identified the type (AFSC) and quantity of personnel needed to support a certain number of deploying aircraft. From this information, the team created a comma-delimited file that identified the type and quantity of personnel needed for each of the different MDS/quantity modeled in the AEF society. The database load program was then used to create the needed personnel records for each of the squadrons modeled.

UTC Database

The UTC database (Figure 33) provides the equipment list that a particular squadron needs when deploying. Currently, only the UTC and UTCDetail tables are being used. For each squadron organization, there is an entry in the UTC table that identifies the name of the organization. This record also identifies a UTCID, which is a key into the UTCDetail table. The UTCDetail table identifies the equipment needed for the deployment. This is accomplished by identifying the NSN and quantity. A current limitation to the UTC database is that it cannot support multiple UTCs for a given organization. This will be changed in the next iteration of the UTC database.

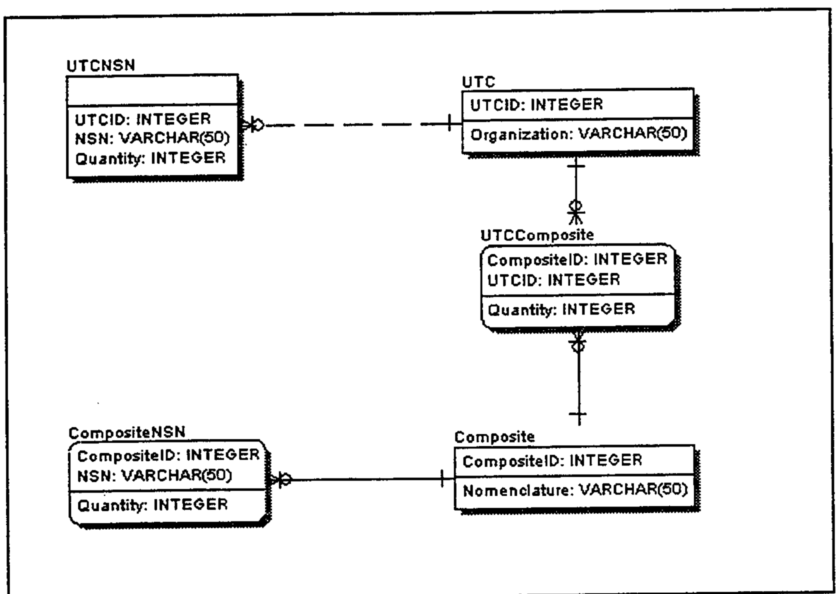


Figure 33: UTC Database Schema

To populate the UTC database, the team utilized UTCs obtained by AFRL/HESR. AFRL/HESR obtained a UTC for a subset of the squadrons modeled in the AEF society. The team then took this subset and determined which of the UTCs would best fit the remaining organizations. The first step in this process was to try and match the MDS to each of the squadron organizations. As with the other databases, the team took the UTCs supplied by HESR and created a series of comma-separated files. These files were then parsed by the database load program to create the data into the UTC database.

UTCPersonnel Database

Similar to the UTC database, the UTCPersonnel database (Figure 34) identifies the type and quantity of personnel needed to deploy based on the number and type of MDS being deployed. For a given combination of aircraft type, aircraft quantity, and squadron name a series of DeploymentCriteria records are created that identifies the quantity of each AFSC needed. The AFSC table identifies the description of each different AFSC represented.

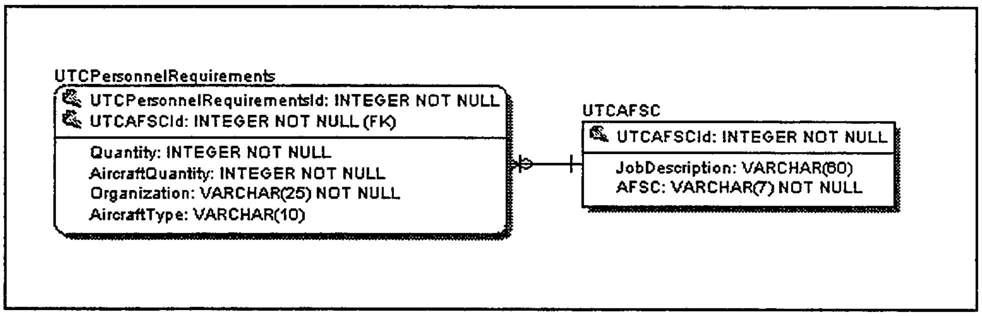


Figure 34: UTCPersonnel Database Schema

The UTCPersonnel database was populated in the same manner as was the UTC database described above.

External Database Connections

The ALP-AEF society has external database interfaces to the Survey Tool for Employment Planning (STEP) and to a database maintained at BBN containing information on fuel burns rates and critical spare part failure rates. The data contained on the database at BBN was obtained from DLA and is referred to as the DLA database among the ALP development team.

The STEP database encompasses a vast amount of site data. For the purposes of the ALP AEF demonstration, only the basic site, airfield, pavement, runway, and taxiway information is accessed.

In order to be able to run standalone, the needed data from the DLA database was modeled in a local database. The needed tables were identified and replicated locally. Figure 35 shows the schema of the locally modeled DLA database.

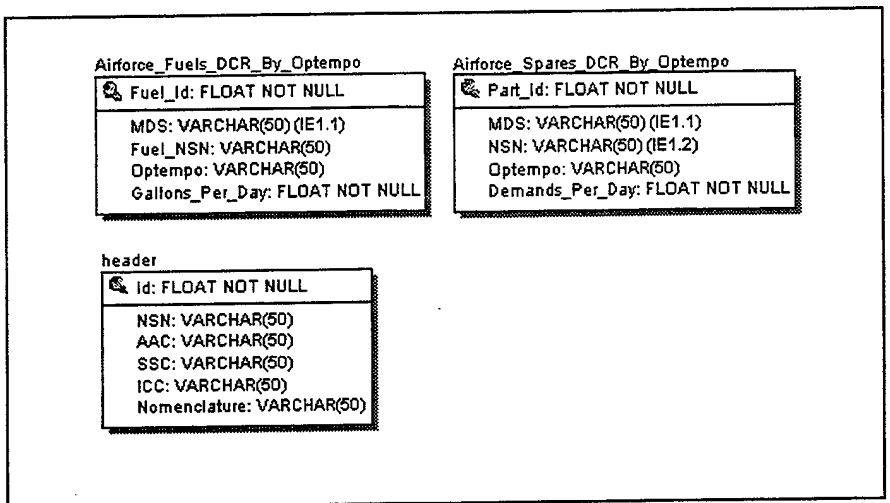


Figure 35: DLA Database Schema

Conclusion

Building on the success of the previous AEF cluster society development task, the government/contractor team successfully added new features and enhancements to the society. Several key accomplishments of the effort included: greatly enhanced cluster society, adding execution monitoring and dynamic replanning, sourcing logic, rule based tailoring logic, and integration with TRANSCOM and DLA cluster societies.

The AEF society grew from 10 clusters to a society consisting of 45 clusters and 266 plugins. Over 14,000 task instances were created and routed through the cluster society. These statistics emphasize a significant increase in the size and complexity of the AEF society over what was developed last year. In addition, the team feels this further indicates that the ALP architecture has the capacity to scale and address more complex decision-making processes.

As with last year's development, the team experienced challenges maintaining compatibility with the evolving ALP infrastructure. Although these challenges are to be expected, the ALPINE team made many changes to their development process, which resulted in a much-improved build release process. Some of these changes included a distributed bug submission and tracking system and a nightly build process. With the addition of these changes and a better intergroup communication, the team was able to more rapidly support new infrastructure builds.

The results of this effort show that there is great promise in the ALP architecture for building a new generation of USAF logistics information systems. The ALP architecture allows clusters to be built to communicate with existing contemporary systems and databases (e.g., Consolidated Aircraft Maintenance System (CAMS)), without redeveloping entire systems. The benefits of the architecture allow an integrated logistics system to be built out of "pluggable" new or existing systems at different command echelons. By complying with the architecture, the integrated system can provide almost seamless transmission of required tasks and data to organizations that need them. The bottlenecks of phone and even E-mail inquiries will be radically reduced if the ALP vision of intelligent decision making clusters is realized. In the relatively simple demonstration developed under this effort, the deployment-planning scenario showed the benefits of the automated tasking and resource allocations that could occur between cooperating organizations using the cluster architecture. One remaining question is whether the architecture is globally scalable. The answer to this question will be driven not only by logistics data requirements at different levels, but by the evolution of network technology for mobile and deployed units. As the bandwidth increases with these systems, it will become more likely that a scalable system can be built. An important observation to make is that the ALP cluster system does not have to solve every logistics problem to be a success. Many decisions could be delegated to the automated evaluation of penalty values, while those considered too critical or complex could be left to human decision-makers. There may be a subset of logistics tasks and data that would fit within the globally available bandwidth and still provide significantly advanced planning speed and quality over what is possible with today's systems.

References

Allen, Christopher S., Patrick K. Clark, Nicholas J. Stute, and Christopher K. Curtis. "Development of an Air Force Wing Level Logistics Cluster for use with the Advanced Logistics Project Architecture." AFRL-HE-WP-TR-1999-0225, March 1999.

ALPINE Joint Venture Document. "Advanced Logistics Program (ALP) PlugIn Developers' Guide" Version ALP-PDG 6.2.

Appendix A

AEF Database Schema